

**This Page is Inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record**

**BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☒ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** \_\_\_\_\_

**IMAGES ARE BEST AVAILABLE COPY.**

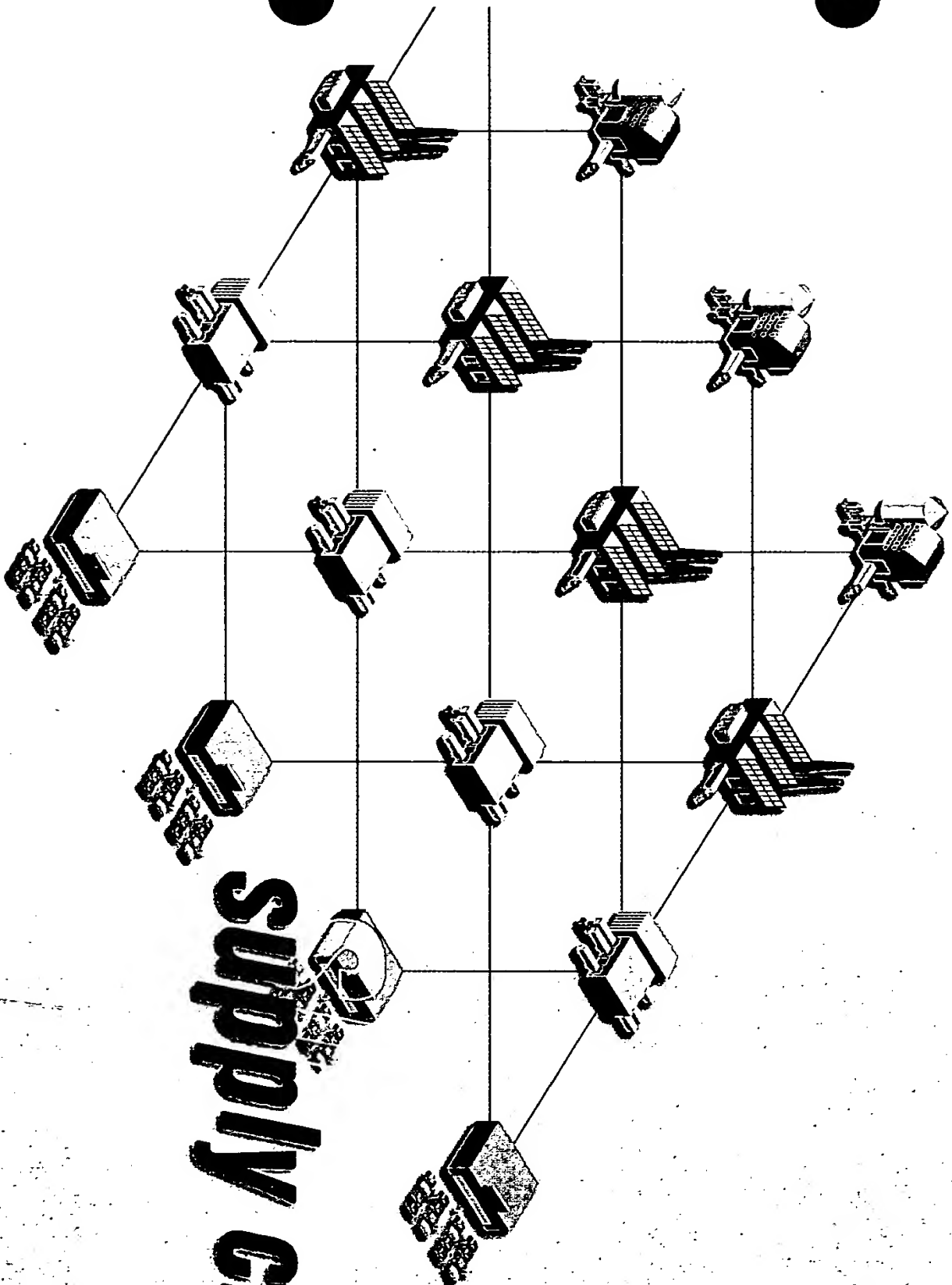
**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**



RHYTHM®

MODEL REFERENCE MANUAL

*intelligent decision support solutions for supply chain planning and optimization*



**Supply chain planner**

i2 Technologies *i2*





**RHYTHM**  
**Supply Chain Planner**  
**Model Reference Manual**

Copyright © 1998  
i2 Technologies, Inc.  
All rights reserved

This notice is intended as a precaution against inadvertent publication and does not imply publication or any waiver of confidentiality. The year included in the foregoing notice is the year of creation of the work.

Information in this document is subject to change without notice and does not represent a commitment on the part of i2 Technologies. The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of the agreement. It is against the law to copy the software on any medium except as specifically allowed in the license or nondisclosure agreement. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage or retrieval systems, for any purpose other than the purchaser's personal use without the express written permission of i2 Technologies.

The information and/or drawings set forth in this document and all rights in and to disclosing or employing the materials, methods, or techniques described herein are the exclusive property of i2 Technologies, Inc.

Unless otherwise noted, all names of companies, products, street addresses, and persons contained herein are part of a completely fictitious scenario or scenarios and are designed solely to document the use of an i2 Technologies product.

© 1998 i2 Technologies, Inc. All rights reserved. Printed in the United States of America. No part of this document may be reproduced in any form, by physical, microfilm, xerography, or any other means, or incorporated into any information retrieval system, electronic or mechanical, without the written permission of i2 Technologies, Inc.

The X Window System is a trademark of the Massachusetts Institute of Technology.

UNIX and Unix are registered trademarks of AT&T.

OIL, Constraint Anchored Optimization, CAO, and RHYTHM are trademarks of i2 Technologies, Inc.

i2 logo and RHYTHM logo are trademarks of i2 Technologies, Inc.

This manual was written, illustrated, and produced with the X/Motif and Windows NT FrameMaker document publishing software.

Written by the development group of i2 Technologies, Inc.

Version 3.07

**i2 TECHNOLOGIES, INC.**  
909 East Las Colinas Blvd.  
16th Floor  
Irving, Texas 75039  
USA

March 26, 1998

## Revision History

<u>Edition</u>	<u>Date</u>	<u>Reason for Revision</u>
3.04Bc1a	09/16/96	Bc1a Release
3.04A	10/14/96	Production Release
3.05Bc1a	12/20/96	Bc1a Release
3.05A	01/31/97	Production Release
3.06Bc1a	10/20/97	Bc1a Release
3.06A	11/10/97	Production Release
3.07Bc1a	03/03/98	Bc1a Release
3.07A	03/27/98	Production Release

---

## Table of Contents

---

1	Introduction.....	25
1.1	Preface.....	25
1.2	Overview.....	25
1.3	RHYTHM Manuals.....	26
1.4	Modeling.....	27
1.4.1	Expressions.....	27
1.4.2	Types.....	27
1.4.3	Fields.....	27
1.4.4	Models.....	28
1.4.5	Submodels.....	28
1.4.6	Extensions.....	29
1.4.7	User-Defined Fields.....	29
1.5	Manual Organization.....	30
1.5.1	HyperText.....	30
1.5.2	Notation.....	30
1.5.3	Chapters.....	30
1.5.4	Fields.....	30
1.5.5	Models.....	30
1.5.6	Extensions.....	31
2	Basic Types.....	33
2.1	Cell_Sluc.....	33
2.2	Computed_String.....	33
2.3	Date.....	33
2.4	Date_Range.....	34
2.5	Event.....	34
2.6	Expression.....	34
2.7	Func.....	34
2.8	Horizon_Date.....	35
2.9	ID.....	35
2.10	Integer.....	35
2.11	Logical.....	35
2.12	Measure.....	35
2.13	Measure_Unit.....	36
2.14	Money.....	36
2.15	Number.....	37
2.16	Palnamic.....	37
2.17	Percentage.....	37
2.18	Predicac.....	37
2.19	Priority.....	37
2.20	Property_List.....	37

Contents

2.21	Quantity.....	37
2.22	Quantity_Range.....	38
2.23	Record.....	38
2.24	Reference.....	38
2.25	Retention.....	38
2.26	Safety_Stock_Unit.....	39
2.27	String.....	39
2.28	Symbol.....	39
2.29	Time.....	39
2.30	Type.....	40
2.31	Typed_Value.....	40
2.32	Void.....	40
3	Basic Functions.....	41
3.1	Operators.....	41
3.2	List Functions.....	44
3.2.1	count ( ListVoid ) .....	45
3.2.2	filter ( ListVoid, Expression, Integer ) .....	45
3.2.3	for_each ( ListVoid, Expression ) .....	46
3.2.4	sort ( ListVoid, Expression ) .....	46
3.2.5	sort_stable ( ListVoid, Expression ) .....	47
3.2.6	sublist ( ListVoid, Integer, Integer ) .....	48
3.2.7	unique ( ListVoid ) .....	48
3.2.8	contains ( ListVoid, Void ) .....	49
3.2.9	found ( ListVoid, Void ) .....	49
3.2.10	element ( ListVoid, Integer ) .....	50
3.2.11	find ( ListVoid, Void ) .....	50
3.2.12	find_or_nonexistent ( ListVoid, Void ) .....	50
3.2.13	find_or_create ( ListVoid, Void ) .....	51
3.2.14	list_index ( ListVoid, Void ) .....	51
3.2.15	first ( ListVoid ) .....	52
3.2.16	last ( ListVoid ) .....	52
3.2.17	list ( ) .....	52
3.2.18	integer ( Integer, Integer ) .....	53
3.3	Mult_Keyed_List Functions.....	53
3.3.1	key ( Symbol, Symbol, Logical, Logical ) .....	54
3.3.2	mult_key ( Symbol, Void ) .....	54
3.3.3	mult_keyed_list ( ListVoid, ListVoid, Expression ) .....	55
3.3.4	mult_keyed_element ( ListVoid, Integer, Symbol, Symbol ) .....	55
3.3.5	key_names ( ListVoid ) .....	56
3.3.6	key_values ( ListVoid, Symbol ) .....	56
3.3.7	mult_key_bucketize ( ListVoid, ListVoid, ListDate_Range, Expression, Expression ) .....	56
3.3.8	mult_key_bucketize ( ListVoid, Integer, Date, Symbol, Symbol ) .....	57
3.4	Singly_Keyed_List Functions.....	57
3.4.1	bucketize ( ListVoid, ListDate_Range, Expression, Expression ) .....	58
3.4.2	bucket_list ( ListVoid, Date, Symbol ) .....	59
3.4.3	bucket_list_by_date ( ListVoid, Date ) .....	60
3.4.4	bucket_list_by_key ( ListVoid, Symbol ) .....	61
3.4.5	bucket_symbols ( ListVoid ) .....	62
3.4.6	keyed_list ( ListVoid, Expression ) .....	62
3.4.7	keyed_element ( Symbol ) .....	63

Contents

3.4.8	keys ( ListVoid ) .....	63
3.5	Recurse_List Functions.....	64
3.5.1	recurse ( ListVoid, Expression ) .....	64
3.5.2	recurse_and_urn ( ListVoid, Expression, Expression ) .....	65
3.5.3	current_depth ( Cell ) .....	66
3.6	Text_String Functions.....	66
3.6.1	string ( Void ) .....	66
3.6.2	string ( Void, Symbol ) .....	67
3.6.3	index ( String, String ) .....	67
3.6.4	index ( String, String, Integer ) .....	68
3.6.5	left ( String ) .....	68
3.6.6	left ( String, Integer ) .....	68
3.6.7	left ( String, String ) .....	69
3.6.8	lower ( String ) .....	69
3.6.9	mid ( String, Integer ) .....	69
3.6.10	mid ( String, Integer, Integer ) .....	70
3.6.11	mid ( String, String, Integer ) .....	70
3.6.12	proper ( String ) .....	70
3.6.13	justify ( String, Integer, String ) .....	71
3.6.14	replace ( String, Integer, Integer, String ) .....	71
3.6.15	right ( String, String ) .....	72
3.6.16	right ( String ) .....	72
3.6.17	right ( String, Integer ) .....	72
3.6.18	upper ( String ) .....	73
3.6.19	trim ( String ) .....	73
3.6.20	trim_left ( String ) .....	73
3.6.21	trim_right ( String ) .....	73
3.6.22	code ( String ) .....	74
3.6.23	code ( String, Integer ) .....	74
3.6.24	len ( String ) .....	74
3.6.25	match_count ( String, String ) .....	75
3.6.26	exact ( String, String ) .....	75
3.6.27	wildcard ( String, String ) .....	76
3.6.28	logical ( String ) .....	76
3.6.29	logical ( String, Symbol ) .....	76
3.6.30	ichar ( Integer ) .....	77
3.6.31	exact ( Symbol, Symbol ) .....	77
3.6.32	symbol ( String ) .....	77
3.6.33	symbol ( String, Symbol ) .....	78
3.7	Numeric Functions.....	78
3.7.1	abs ( Integer ) .....	78
3.7.2	abs ( Number ) .....	78
3.7.3	abs ( Quantity ) .....	79
3.7.4	abs ( Time ) .....	79
3.7.5	finite ( Number ) .....	79
3.7.6	convert ( Quantity, Quantity ) .....	80
3.7.7	cos ( Number ) .....	80
3.7.8	exp ( Number ) .....	80
3.7.9	integer ( Number ) .....	81
3.7.10	integer ( Percentage ) .....	81

## Contents

3.7.11	integer (String).....	81
3.7.12	integer (String, Symbol).....	82
3.7.13	in (Number).....	82
3.7.14	log (Number).....	82
3.7.15	log (Number, Number).....	82
3.7.16	log10 (Number).....	83
3.7.17	measure (Quantity).....	83
3.7.18	measure (String).....	83
3.7.19	measure (String, Symbol).....	84
3.7.20	money (String).....	84
3.7.21	money (String, Symbol).....	85
3.7.22	number (Integer).....	85
3.7.23	number (Percentage).....	85
3.7.24	number (Quantity).....	85
3.7.25	number (String).....	86
3.7.26	number (String, Symbol).....	86
3.7.27	percentage (Integer).....	86
3.7.28	percentage (Number).....	87
3.7.29	percentage (Quantity).....	87
3.7.30	percentage (String).....	87
3.7.31	percentage (String, Symbol).....	88
3.7.32	power (Number, Number).....	88
3.7.33	quantity (String).....	88
3.7.34	quantity (Integer).....	88
3.7.35	quantity (Number).....	89
3.7.36	quantity (String).....	89
3.7.37	quantity (String, Symbol).....	90
3.7.38	quantity (Time).....	90
3.7.39	round (Number).....	90
3.7.40	round (Number, Number).....	90
3.7.41	round_down (Number).....	91
3.7.42	round_down (Number, Number).....	91
3.7.43	round_in (Number).....	91
3.7.44	round_in (Number, Number).....	92
3.7.45	round_out (Number).....	92
3.7.46	round_out (Number, Number).....	92
3.7.47	round_up (Number).....	93
3.7.48	round_up (Number, Number).....	93
3.7.49	sin (Number).....	94
3.7.50	sqr (Number).....	94
3.7.51	tan (Number).....	94
3.7.52	time (Quantity).....	94
3.7.53	time (String).....	95
3.7.54	time (String, Symbol).....	95
3.7.55	units (Quantity).....	96
3.8	Random Functions.....	96
3.8.1	rand_integer (Integer, Integer).....	96
3.8.2	rand ( ).....	96
3.8.3	rand (Number, Number).....	97
3.8.4	rand_seed (Number).....	97

## Contents

3.9	Quantity_Range Functions.....	97
3.9.1	intersects (Quantity_Range, Quantity_Range).....	97
3.9.2	within (Quantity, Quantity_Range).....	98
3.9.3	within (Quantity_Range, Quantity_Range).....	99
3.9.4	interval (Quantity_Range).....	99
3.9.5	limit (Quantity, Quantity_Range).....	100
3.9.6	limit (Quantity_Range, Quantity_Range).....	100
3.9.7	max (Quantity_Range).....	101
3.9.8	min (Quantity_Range).....	101
3.9.9	enclose (Quantity, Quantity_Range).....	102
3.9.10	enclose (Quantity_Range, Quantity_Range).....	102
3.9.11	quantity_interval (Quantity, Quantity).....	103
3.9.12	quantity_range (Quantity, Quantity).....	103
3.9.13	quantity_range (String).....	104
3.9.14	quantity_range (String, Symbol).....	104
3.9.15	sel_max (Quantity_Range, Quantity).....	105
3.9.16	sel_min (Quantity_Range, Quantity).....	105
3.9.17	sel_interval (Quantity_Range, Quantity).....	106
3.10	Numeric_List Functions.....	106
3.10.1	average (List(Integer)).....	106
3.10.2	average (List(Integer)).....	107
3.10.3	average (List(Percentage)).....	107
3.10.4	average (List(Quantity)).....	107
3.10.5	average (List(Time)).....	108
3.10.6	max (List(Integer)).....	108
3.10.7	max (List(Integer)).....	108
3.10.8	max (List(Percentage)).....	109
3.10.9	max (List(Quantity)).....	109
3.10.10	max (List(Time)).....	109
3.10.11	min (List(Integer)).....	109
3.10.12	min (List(Integer)).....	110
3.10.13	min (List(Percentage)).....	110
3.10.14	min (List(Quantity)).....	110
3.10.15	min (List(Time)).....	111
3.10.16	product (List(Integer)).....	111
3.10.17	product (List(Integer)).....	111
3.10.18	product (List(Percentage)).....	112
3.10.19	product (List(Quantity)).....	112
3.10.20	product (List(Integer)).....	112
3.10.21	slice (List(Integer)).....	113
3.10.22	slice (List(Percentage)).....	113
3.10.23	slice (List(Quantity)).....	113
3.10.24	slice (List(Time)).....	114
3.10.25	sum (List(Integer)).....	114
3.10.26	sum (List(Integer)).....	114
3.10.27	sum (List(Percentage)).....	115
3.10.28	sum (List(Quantity)).....	115
3.10.29	sum (List(Time)).....	115
3.11	Date Functions.....	115
3.11.1	max (List(Date)).....	116

Contents

3.11.2	min (List(Date))	116
3.11.3	date (String)	116
3.11.4	date (String, Symbol)	117
3.11.5	now ( )	117
3.11.6	start_of_day (Date)	118
3.11.7	start_of_day (Date, Integer)	118
3.11.8	start_of_month (Date)	118
3.11.9	start_of_month (Date, Integer)	119
3.11.10	start_of_week (Date)	119
3.11.11	start_of_week (Date, Integer)	119
3.11.12	start_of_year (Date)	120
3.11.13	start_of_year (Date, Integer)	120
3.11.14	horizon_date (String)	120
3.11.15	finite (Date)	122
3.11.16	day_of_month (Date)	122
3.11.17	day_of_week (Date)	122
3.11.18	day_of_year (Date)	122
3.11.19	hour (Date)	123
3.11.20	minute (Date)	123
3.11.21	month (Date)	123
3.11.22	number_of_weeks (Date, Date)	124
3.11.23	second (Date)	124
3.11.24	week_of_year (Date)	124
3.11.25	year (Date)	124
3.11.26	within_daylight_savings_time (Date)	125
3.11.27	within_leap_year (Date)	125
3.11.28	enclosing_day (Date)	125
3.11.29	date (Horizon, Date, Date)	126
3.11.30	restriction (String)	126
3.11.31	restriction (String, Date)	126
3.11.32	restriction (String, Date, Range)	126
3.11.33	satisfies (Restriction, Date, Range)	127
3.12	Date_Range Functions	127
3.12.1	date_range (Date, Date)	127
3.12.2	date_range (Date, Time)	127
3.12.3	date_range (String)	128
3.12.4	date_range (String, Symbol)	128
3.12.5	enclose (Date, Date, Range)	128
3.12.6	enclose (Date_Range, Date_Range)	129
3.12.7	end (Date, Range)	129
3.12.8	intersect (Date, Range, Date_Range)	130
3.12.9	limit (Date, Date, Range)	131
3.12.10	limit (Date_Range, Date, Range)	131
3.12.11	set_end (Date_Range, Date)	132
3.12.12	set_start (Date, Range, Date)	132
3.12.13	set_time (Date, Range, Time)	133
3.12.14	start (Date, Range)	133
3.12.15	time (Date, Range)	134
3.12.16	within (Date, Date, Range)	134
3.12.17	within (Date, Range, Date, Range)	135

Contents

3.13	Date_Range_List Functions	135
3.13.1	days (Date_Range)	135
3.13.2	days (Date_Range, Time)	136
3.13.3	months (Date_Range)	137
3.13.4	quarters (Date_Range)	138
3.13.5	shifts (Date_Range)	138
3.13.6	shifts (Date, Range, Time)	139
3.13.7	shifts (Date_Range, Time, Time)	140
3.13.8	weeks (Date_Range)	141
3.13.9	weeks (Date_Range, Integer)	142
3.14	Logical Functions	143
3.14.1	and (Logical, Logical)	143
3.14.2	not (Logical)	143
3.14.3	or (Logical, Logical)	144
3.14.4	xor (Logical, Logical)	144
3.15	Miscellaneous Functions	145
3.15.1	id (Void)	145
3.15.2	current_index (Cell)	145
3.15.3	system (String)	146
3.15.4	values (Type)	146
3.15.5	data (Symbol)	146
3.15.6	exists (Void)	146
3.15.7	exists_without_errors (Void)	147
3.15.8	export (string, void)	147
3.15.9	import (string, void)	147
3.15.10	engine_name ( )	148
3.15.11	get_color (String)	148
3.15.12	get_drag_string (Integer)	149
3.15.13	getenv (String)	149
3.15.14	translate (String)	149
3.15.15	model_file ( )	149
3.15.16	option (String)	149
3.15.17	scenev (String, String)	150
3.15.18	typed_value (void)	150
3.15.19	owner (Typed_Value)	151
3.15.20	first_value (Typed_Value)	151
3.15.21	delete (Void)	151
3.15.22	make_type (Void, Type)	151
3.15.23	nonexistent ( )	152
3.15.24	set_option (String, String)	152
3.15.25	type (Typed_Value)	152
3.15.26	type (String)	152
3.15.27	model_type (void)	153
3.15.28	value (Typed_Value, Type)	153
3.15.29	trace_time ( )	153
3.15.30	memory_size ( )	154
3.15.31	process_size ( )	154
3.15.32	with_connection (Connection, void)	155
3.15.33	functions (Symbol)	155
3.15.34	echo (Logical)	156

## Contents

3.15.35	echo (String).....	156
3.15.36	with_echo_file (String, void).....	156
3.16	File Functions.....	156
3.16.1	is_directory (String).....	156
3.16.2	is_file (String).....	157
3.16.3	is_writable (String).....	157
3.17	Evaluation Functions.....	157
3.17.1	background (Expression).....	157
3.17.2	do ( ).....	158
3.17.3	call (Symbol).....	158
3.17.4	do_file_echo (String).....	158
3.17.5	do_file (String).....	159
3.17.6	do_file_fast (String).....	159
3.17.7	engine (Expression).....	159
3.17.8	evaluate (Expression).....	160
3.17.9	if (Logical, Void, Void).....	160
3.17.10	while (Logical, Void).....	160
3.17.11	reference (Variable).....	161
3.18	RhythmLink Functions.....	161
3.18.1	rl_field (Data_Table, String).....	161
3.18.2	rhythmLink_field (Symbol, String).....	162
3.18.3	rl_record_field (Record, String).....	162
3.19	Debug Functions.....	162
3.19.1	stop (Symbol, Symbol).....	163
3.19.2	stop (Symbol).....	163
3.19.3	break (Expression, Symbol, Expression).....	163
3.19.4	breakpoint ( ).....	164
3.19.5	breakpoint (Symbol).....	164
3.19.6	disable ( ).....	164
3.19.7	disable (Symbol).....	164
3.19.8	enable ( ).....	164
3.19.9	enable (Symbol).....	164
3.19.10	next ( ).....	164
3.19.11	step ( ).....	164
3.19.12	cont ( ).....	164
3.19.13	next (Integer).....	164
3.19.14	step (Integer).....	165
3.19.15	cont (Integer).....	165
3.19.16	watch (Expression, Symbol, Expression).....	165
3.19.17	unwatch (Symbol).....	165
3.19.18	where ( ).....	165
3.19.19	wherets ( ).....	165
3.19.20	print_variables ( ).....	165
3.19.21	print_report_variables ( ).....	166
3.19.22	inspect (void).....	166
3.19.23	inspectS (Integer).....	166
3.19.24	inspectH (Integer).....	166
3.19.25	inspectHS (Integer, Integer).....	166
3.19.26	inspectV (Integer).....	167
3.19.27	inspectV (Integer, Integer).....	167

## Contents

3.20	Program Functions.....	167
3.20.1	user ( ).....	167
3.20.2	quit (String).....	167
3.20.3	shutdown (String).....	168
3.21	Engine Queue Functions.....	168
3.21.1	wait ( ).....	168
4	Summary Section.....	169
5	Models.....	228
5.1	Supply Chain Model.....	232
5.1.1	Sic Model.....	235
5.1.1.1	Standard Extensions of model Sic.....	242
5.1.1.1.1	role extensions of model Sic.....	242
5.1.1.1.1.1	LINK Extension.....	242
5.1.1.1.1.2	SUPPLIER Extension.....	244
5.1.1.1.1.3	CUSTOMER Extension.....	245
5.1.1.2	role submodels of model Sic.....	246
5.1.1.3	Location Model.....	246
5.1.1.4	licm Model.....	249
5.1.1.4.1	spec submodels of model licm.....	254
5.1.1.5	Buffer Model.....	254
5.1.1.5.1	Buffer_Problem_Detector Model.....	260
5.1.1.5.2	flow_policy submodels of model Buffer.....	262
5.1.1.5.3	Flow_Criterion Model.....	262
5.1.1.6	Resource Model.....	263
5.1.1.6.1	Resource_Skill Model.....	270
5.1.1.6.2	efficiency submodels of model Resource.....	271
5.1.1.6.3	size submodels of model Resource.....	272
5.1.1.6.4	Size_Dimension Model.....	272
5.1.1.6.5	load_policy submodels of model Resource.....	273
5.1.1.6.6	Resource_Setup_Order Model.....	273
5.1.1.6.7	Resource_Blocks Model.....	273
5.1.1.7	Skill Model.....	273
5.1.1.7.1	Skill_Resource Model.....	275
5.1.1.8	Operation Model.....	276
5.1.1.8.1	Load Model.....	288
5.1.1.8.1.1	Load_Size Model.....	291
5.1.1.8.2	Flow Model.....	292
5.1.1.8.2.1	usage_policy submodels of model Flow.....	294
5.1.1.8.3	Operation_Problem_Detector Model.....	295
5.1.1.8.4	process submodels of model Operation.....	297
5.1.1.8.5	Alternate_Operation Model.....	297
5.1.1.8.6	Effective_Calendar_Operation Model.....	298
5.1.1.8.7	Routing_Operation Model.....	299
5.1.1.9	Configuration Model.....	300
5.1.1.9.1	spec submodels of model Configuration.....	302
5.1.1.10	licm_Group Model.....	302
5.1.1.10.1	Sub_licm_Group Model.....	305
5.1.1.10.2	Sub_licm Model.....	306
5.1.2	Seller Model.....	307
5.1.2.1	Sic_Group Model.....	313

---

**Contents**


---

5.1.2.1.1	Explicit_Site Model	315
5.1.2.2	Product_Reot Model	316
5.1.2.2.1	Product_Supplier Model	320
5.1.2.2.1.1	Product_Item Model	321
5.1.2.3	Product Model	322
5.1.2.3.1	Generic_Product Model	334
5.1.2.3.2	Allocname_Product Model	335
5.1.2.3.3	allocation_policy submodels of model Product	336
5.1.2.3.4	Product_Allocation Model	336
5.1.2.4	Product_Group Model	337
5.1.2.4.1	Sub_Product_Group Model	341
5.1.2.4.2	Sub_Product Model	343
5.1.3	Plan Model	345
5.1.3.1	Site_Plan Model	350
5.1.3.1.1	Standard Extensions of model Site_Plan	352
5.1.3.1.1.1	role extensions of model Site_Plan	352
5.1.3.1.1.2	LINK Extension	352
5.1.3.1.1.3	SUPPLIER Extension	355
5.1.3.1.2	CUSTOMER Extension	355
5.1.3.1.3	role submodels of model Site_Plan	356
5.1.3.1.3.1	Buffer_Plan Model	356
5.1.3.1.3.2	Lot Model	361
5.1.3.1.3.3	flow_policy submodels of model Buffer_Plan	363
5.1.3.1.4	Sorted_Bucket Model	363
5.1.3.1.4.1	Resource_Plan Model	364
5.1.3.1.4.2	load_policy submodels of model Resource_Plan	373
5.1.3.1.5	Consolidation Model	373
5.1.3.1.5.1	Operation_Plan Model	381
5.1.3.1.5.2	Load_Plan Model	383
5.1.3.1.5.2.1	Flow_Plan Model	385
5.1.3.1.6	Lot_Flow Model	385
5.1.3.1.7	Operation_Site Model	386
5.1.3.1.7.1	Request Model	388
5.1.3.1.7.1.1	Delivery_Request Model	397
5.1.3.1.8	Item_Request Model	405
5.1.3.1.8.1	Promise Model	411
5.1.3.1.8.1.1	Delivery_Promise Model	416
5.1.3.1.8.1.2	Delivery_Available_To_Promise Model	424
5.1.3.1.8.1.2.1	Item_Promise Model	426
5.1.3.1.8.1.2.1.1	Item_Available_To_Promise Model	433
5.1.3.1.8.1.2.1.1.1	Product_Available_To_Promise Model	435
5.1.3.1.9	Acceptance Model	437
5.1.3.1.9.1	Delivery_Acceptance Model	440
5.1.3.1.9.1.1	Item_Acceptance Model	444
5.1.3.2	Seller_Plan Model	449
5.1.3.2.1	Forecast Model	453
5.1.3.2.1.1	Standard Extensions of model Forecast	459
5.1.3.2.1.1.1	level extensions of model Forecast	459
5.1.3.2.1.1.1.1	INDIVIDUAL Extension	459
5.1.3.2.1.1.1.1.1	GROUP Extension	459

---

**Contents**


---

5.1.3.2.1.2	Forecast_Entry Model	461
5.1.3.2.1.3	level submodels of model Forecast	478
5.1.3.2.1.4	ATP_Entry Model	478
5.1.3.3	Problem Model	480
5.1.3.4	Active_Strategy Model	483
5.1.3.4.1	Active_Problem Model	488
5.1.3.4.2	Active_Goal Model	490
5.2	Problem_Category Model	492
5.3	Horizon Model	494
5.3.1	bucket_spec submodels of model Horizon	496
5.3.2	Horizon_Bucket_Start Model	496
5.4	Strategy Model	497
5.4.1	Problem_Set Model	502
5.4.2	Strategy_Change Model	505
5.4.3	Strategy_Lock Model	507
5.4.4	Strategy_Goal Model	508
5.4.5	execution submodels of model Strategy	509
5.4.6	Ordered_Sub_Strategy Model	509
5.4.7	Ranked_Sub_Strategy Model	509
5.5	Unit Model	511
5.5.1	Unit_Quantity Model	513
5.6	Profile_Number Model	514
5.7	Profile_Percentage Model	515
5.8	Profile_Quantity Model	516
5.9	Box Model	517
5.10	Calendar Model	519
5.10.1	Calendar_Entry Model	522
5.10.2	Sub_Calendar Model	528
5.11	Calendar_Plan Model	530
6	Control Model	531
7	Connection Model	531
7.1	User Model	532
7.1.1	Report_Directory Model	536
7.1.2	Report Model	538
7.1.3	Layout Model	539
7.1.4	Worksheet Model	540
7.1.5	Style Model	541
7.1.6	Format Model	542
7.1.7	Domain Model	544
7.2	Model_Type Model	545
7.2.1	Field Model	547
7.2.2	Extension_Selector Model	550
7.3	Field_Error Model	551
8	Function Model	553
9	Breakpoint Model	554
10	Extensions	555
10.1	Site Extensions	568
10.1.1	role extensions of model Site	568
10.1.1.1	LINK Extension	568
10.1.1.2	SUPPLIER Extension	570



## Contents

10.1.1.3	CUSTOMER Extension.....	570
10.2	Seller Extensions.....	572
10.2.1	request_naming extensions of model Seller .....	572
10.2.1.1	NONE Extension.....	572
10.3	Site_Group Extensions .....	573
10.3.1	spec extensions of model Site_Group.....	573
10.4	Product_Root Extensions .....	574
10.4.1	forecast_policy extensions of model Product_Root.....	574
10.4.1.1	SIMPLE_FIXED_QUANTITY Extension.....	574
10.4.1.2	SINGLE_REQUEST Extension.....	575
10.4.1.3	SIMPLE_FIXED_TIME Extension.....	575
10.4.1.4	WEEKLY Extension.....	577
10.4.1.5	DUAL_REQUEST Extension.....	578
10.5	Product Extensions .....	580
10.5.1	forecast_policy extensions of model Product.....	580
10.5.1.1	SIMPLE_FIXED_QUANTITY Extension.....	580
10.5.1.2	SINGLE_REQUEST Extension.....	580
10.5.1.3	SIMPLE_FIXED_TIME Extension.....	580
10.5.1.4	WEEKLY Extension.....	580
10.5.1.5	DUAL_REQUEST Extension.....	580
10.5.2	expiration_policy extensions of model Product.....	580
10.5.2.1	AT_END Extension.....	580
10.5.3	allocation_policy extensions of model Product .....	580
10.5.3.1	FCFS Extension.....	580
10.5.3.2	PER_ALLOCATED Extension.....	581
10.5.3.3	PER_COMMITTED Extension.....	581
10.5.3.4	MEMBER_RANK Extension.....	581
10.5.3.5	FIXED_SPLIT Extension.....	583
10.5.4	availability_policy extensions of model Product .....	584
10.5.4.1	SLIDING Extension.....	584
10.5.4.2	HORIZON Extension.....	584
10.5.5	price_policy extensions of model Product.....	585
10.5.5.1	NONE Extension.....	585
10.5.5.2	FIXED Extension.....	585
10.6	Operation Extensions .....	586
10.6.1	process extensions of model Operation .....	586
10.6.1.1	ALTERNATES_PRIMARY Extension.....	586
10.6.1.2	ALTERNATES_PROPORTIONAL Extension.....	586
10.6.1.3	EFFECTIVE_CALENDAR Extension.....	587
10.6.1.4	DELAY_ONLY_FIXED Extension.....	588
10.6.1.5	DELAY_ONLY_BASIC Extension.....	588
10.6.1.6	BASIC_CALENDARS Extension.....	588
10.6.1.7	FIXED_TIME Extension.....	589
10.6.1.8	TIME_MULTIPLE Extension.....	590
10.6.1.9	BASIC Extension.....	590
10.6.1.10	BASIC_DELAYED Extension.....	590
10.6.1.11	REQUEST_FIXED Extension.....	591
10.6.1.12	REQUEST_FIXED_WITH_ANALYSIS Extension.....	593
10.6.1.13	ROUTING Extension.....	594
10.7	Load Extensions .....	596

## Contents

10.7.1	usage_policy extensions of model Load .....	596
10.7.1.1	RESOURCE Extension.....	596
10.7.1.2	ONE Extension.....	596
10.8	Load_Size Extensions .....	597
10.8.1	load_size_usage extensions of model Load_Size .....	597
10.8.1.1	FIXED Extension.....	597
10.8.1.2	LINEAR Extension.....	597
10.9	Flow Extensions.....	598
10.9.1	usage_policy extensions of model Flow .....	598
10.9.1.1	CONSUME_FIXED Extension.....	598
10.9.1.2	CONSUME_PER Extension.....	598
10.9.1.3	PRODUCE_FIXED Extension.....	598
10.9.1.4	PRODUCE_PER Extension.....	599
10.9.1.5	PRODUCE_YIELD Extension.....	599
10.9.1.6	PRODUCE_YIELD_CALENDAR Extension.....	600
10.10	Operation_Problem_Detector Extensions.....	602
10.10.1	detector extensions of model Operation_Problem_Detector.....	602
10.11	Operation_Plan Extensions .....	603
10.11.1	motive extensions of model Operation_Plan.....	603
10.11.1.1	PRODUCE Extension.....	603
10.11.1.2	CONSUME Extension.....	603
10.11.1.3	DELIVER Extension.....	604
10.11.1.4	RECEIVE Extension.....	605
10.11.2	process extensions of model Operation_Plan .....	605
10.11.2.1	ALTERNATES_PRIMARY Extension.....	605
10.11.2.2	ALTERNATES_PROPORTIONAL Extension.....	606
10.11.2.3	EFFECTIVE_CALENDAR Extension.....	606
10.11.2.4	DELAY_ONLY_FIXED Extension.....	606
10.11.2.5	DELAY_ONLY_BASIC Extension.....	607
10.11.2.6	BASIC_CALENDARS Extension.....	607
10.11.2.7	FIXED_TIME Extension.....	607
10.11.2.8	TIME_MULTIPLE Extension.....	607
10.11.2.9	BASIC Extension.....	607
10.11.2.10	BASIC_DELAYED Extension.....	607
10.11.2.11	REQUEST_FIXED Extension.....	607
10.11.2.12	REQUEST_FIXED_WITH_ANALYSIS Extension.....	608
10.11.2.13	ROUTING Extension.....	608
10.12	Resource Extensions .....	610
10.12.1	efficiency extensions of model Resource .....	610
10.12.1.1	FIXED Extension.....	610
10.12.1.2	CALENDAR Extension.....	610
10.12.2	variability extensions of model Resource .....	610
10.12.2.1	ZERO Extension.....	610
10.12.2.2	FIXED Extension.....	610
10.12.3	maintenance extensions of model Resource .....	611
10.12.3.1	ZERO Extension.....	611
10.12.4	size extensions of model Resource .....	611
10.12.4.1	UNLIMITED Extension.....	611
10.12.4.2	FIXED_COUNT Extension.....	612
10.12.4.3	FIXED_QUANTITY Extension.....	612

## Contents

10.12.4.4	CALENDAR_COUNT Extension.....	612
10.12.4.5	MULTI_DIMENSION Extension.....	613
10.12.5	load_policy extensions of model Resource.....	614
10.12.5.1	SIMPLE_CONSOLIDATION Extension.....	614
10.12.5.2	INFINITE_USE Extension.....	614
10.12.5.3	EXCLUSIVE_USE Extension.....	614
10.12.5.4	SHARED_USE Extension.....	614
10.13	Resource_Skill Extensions.....	617
10.13.1	efficiency extensions of model Resource_Skill.....	617
10.13.1.1	FIXED Extension.....	617
10.13.1.2	CALENDAR Extension.....	617
10.14	Resource_Problem_Detector Extensions.....	618
10.14.1	detector extensions of model Resource_Problem_Detector.....	618
10.15	Resource_Plan Extensions.....	619
10.15.1	efficiency extensions of model Resource_Plan.....	619
10.15.2	load_policy extensions of model Resource_Plan.....	619
10.15.2.1	SIMPLE_CONSOLIDATION Extension.....	619
10.15.2.2	INFINITE_USE Extension.....	619
10.15.2.3	EXCLUSIVE_USE Extension.....	619
10.15.2.4	SHARED_USE Extension.....	619
10.15.3	size extensions of model Resource_Plan.....	620
10.15.4	maintenance extensions of model Resource_Plan.....	620
10.16	Buffer Extensions.....	621
10.16.1	flow_policy extensions of model Buffer.....	621
10.16.1.1	BUCKETED_NESTED_SORT Extension.....	621
10.16.1.2	PRODUCING_FLOW_CALENDAR Extension.....	623
10.16.1.3	PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK Extension.....	628
10.16.1.4	SUPPLY_CALENDAR Extension.....	636
10.16.1.5	ON_HAND_CALENDAR Extension.....	637
10.16.1.6	ON_HAND_CALENDAR_FILTER_AND_RANK Extension.....	638
10.16.1.7	FLOW_LIMIT_CALENDAR Extension.....	642
10.16.1.8	FLOW_LIMIT_CALENDAR_FILTER_AND_RANK Extension.....	643
10.16.1.9	INFINITE Extension.....	647
10.16.1.10	SUPPLIER Extension.....	647
10.16.1.11	BASIC Extension.....	648
10.16.1.12	BASIC_FILTER_AND_RANK Extension.....	649
10.16.1.13	FIXED_QUANTITY Extension.....	654
10.16.1.14	MULTIPLE Extension.....	656
10.16.1.15	MULTIPLE_FILTER_AND_RANK Extension.....	657
10.16.1.16	FIXED_QUANTITY_FENCED Extension.....	663
10.16.1.17	FIXED_TIME Extension.....	665
10.16.1.18	LFL_SIMPLE Extension.....	666
10.16.1.19	LFL_BOUNDED Extension.....	667
10.16.1.20	MLFL_BOUNDED Extension.....	669
10.16.2	stocking_policy extensions of model Buffer.....	671
10.16.2.1	MANUAL Extension.....	671
10.16.2.2	CALENDAR Extension.....	671
10.17	Buffer_Problem_Detector Extensions.....	676
10.17.1	detector extensions of model Buffer_Problem_Detector.....	676
10.18	Buffer_Plan Extensions.....	677

## Contents

10.18.1	flow_policy extensions of model Buffer_Plan.....	677
10.18.1.1	BUCKETED_NESTED_SORT Extension.....	677
10.18.1.2	PRODUCING_FLOW_CALENDAR Extension.....	677
10.18.1.3	PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK Extension.....	681
10.18.1.4	ON_HAND_CALENDAR Extension.....	685
10.18.1.5	ON_HAND_CALENDAR_FILTER_AND_RANK Extension.....	685
10.18.1.6	FLOW_LIMIT_CALENDAR Extension.....	686
10.18.1.7	FLOW_LIMIT_CALENDAR_FILTER_AND_RANK Extension.....	686
10.18.1.8	BASIC Extension.....	686
10.18.1.9	BASIC_FILTER_AND_RANK Extension.....	690
10.18.1.10	FIXED_QUANTITY Extension.....	694
10.18.1.11	MULTIPLE Extension.....	697
10.18.1.12	MULTIPLE_FILTER_AND_RANK Extension.....	701
10.18.1.13	FIXED_QUANTITY_FENCED Extension.....	705
10.18.1.14	FIXED_TIME Extension.....	709
10.19	Forecast Extensions.....	713
10.19.1	level extensions of model Forecast.....	713
10.19.1.1	INDIVIDUAL Extension.....	713
10.19.1.2	GROUP Extension.....	713
10.20	Forecast_Entry Extensions.....	715
10.20.1	forecast_policy extensions of model Forecast_Entry.....	715
10.20.2	allocation_policy extensions of model Forecast_Entry.....	715
10.20.2.1	MEMBER_RANK Extension.....	715
10.21	Site_Plan Extensions.....	716
10.21.1	role extensions of model Site_Plan.....	716
10.21.1.1	LINK Extension.....	716
10.21.1.2	SUPPLIER Extension.....	719
10.21.1.3	CUSTOMER Extension.....	719
10.22	Problem Extensions.....	720
10.22.1	category extensions of model Problem.....	720
10.22.1.1	REQUEST_NOT_PLANNED Extension.....	720
10.22.1.2	REQUEST_PLANNED_LATE Extension.....	721
10.22.1.3	REQUEST_PLANNED_EARLY Extension.....	723
10.22.1.4	REQUEST_PLANNED_SHORT Extension.....	723
10.22.1.5	REQUEST_PLANNED_EXCESS Extension.....	726
10.22.1.6	PROMISE_NOT_PLANNED Extension.....	728
10.22.1.7	PROMISE_PLANNED_LATE Extension.....	730
10.22.1.8	PROMISE_PLANNED_EARLY Extension.....	731
10.22.1.9	PROMISE_PLANNED_SHORT Extension.....	733
10.22.1.10	PROMISE_PLANNED_EXCESS Extension.....	734
10.22.1.11	ACCEPTANCE_NOT_PLANNED Extension.....	736
10.22.1.12	ACCEPTANCE_PLANNED_LATE Extension.....	738
10.22.1.13	ACCEPTANCE_PLANNED_EARLY Extension.....	739
10.22.1.14	ACCEPTANCE_PLANNED_SHORT Extension.....	741
10.22.1.15	ACCEPTANCE_PLANNED_EXCESS Extension.....	743
10.22.1.16	UNRELEASED_BEFORE_CURRENT Extension.....	744
10.22.1.17	UNRELEASED Extension.....	745
10.22.1.18	NEEDS_RELEASE Extension.....	745
10.22.1.19	INCONSISTENT_OPPLAN Extension.....	745
10.22.1.20	REQUEST_PROMISED_LATE Extension.....	746

## Contents

10.22.1.21	REQUEST_PROMISED_EARLY Extension	747
10.22.1.22	REQUEST_PROMISED_SHORT Extension	748
10.22.1.23	REQUEST_PROMISED_EXCESS Extension	750
10.22.1.24	ITEM_PROMISE_OVERPRICED Extension	751
10.22.1.25	DELIVERY_PROMISE_OVERPRICED Extension	752
10.22.1.26	PROMISE_NOT_OFFERED Extension	753
10.22.1.27	PROMISE_NOT_ACCEPTED Extension	754
10.22.1.28	ACCEPTANCE_INCONSISTENT Extension	755
10.22.1.29	REQUEST_QUEUED Extension	755
10.22.1.30	DELIVERY_REQUEST_NOT_COORDINATED Extension	756
10.22.1.31	DELIVERY_PROMISE_NOT_COORDINATED Extension	757
10.22.1.32	DELIVERY_ACCEPTANCE_NOT_COORDINATED Extension	758
10.22.1.33	NEGATIVE_ATP Extension	759
10.22.1.34	NEGATIVE_PLANNED_ATP Extension	760
10.22.1.35	OVER_COMMITTED Extension	760
10.22.1.36	OVER_CONSUMED Extension	761
10.22.1.37	UNALLOCATED_FORECAST Extension	761
10.22.1.38	SUPPLY_PLANNED_LATE Extension	762
10.22.1.39	SUPPLY_PLANNED_EARLY Extension	763
10.22.1.40	SUPPLY_PLANNED_SHORT Extension	765
10.22.1.41	SUPPLY_PLANNED_EXCESS Extension	767
10.22.1.42	SUPPLY_PROMISED_LATE Extension	769
10.22.1.43	SUPPLY_PROMISED_EARLY Extension	770
10.22.1.44	SUPPLY_PROMISED_SHORT Extension	772
10.22.1.45	SUPPLY_PROMISED_EXCESS Extension	773
10.22.1.46	UNIDENTIFIED_OP_STATE Extension	775
10.22.1.47	UNCOORDINATED Extension	775
10.22.1.48	UNCOORDINATED Extension	776
10.22.1.49	CONSOLIDATION_OVERSIZE Extension	776
10.22.1.50	CONSOLIDATION_UNDSIZE Extension	777
10.22.1.51	OVERLOAD Extension	778
10.22.1.52	OVERSIZE Extension	778
10.22.1.53	BUCKET_OVERSIZE Extension	779
10.22.1.54	UNDERLOAD Extension	779
10.22.1.55	NEGATIVE_ON_HAND Extension	780
10.22.1.56	OVER_FLOW_LIMIT Extension	781
10.22.1.57	NEGATIVE_ON_HAND_AT_END Extension	781
10.22.1.58	LOT_OVER_CONSUMED Extension	782
10.22.1.59	LOT_NOT_CONSUMED Extension	782
10.22.1.60	LOT_NOT_PRODUCED Extension	783
10.22.1.61	LOT_OVER_PRODUCED Extension	783
10.22.1.62	LOW_ON_HAND Extension	784
10.22.1.63	EXCESS_ON_HAND Extension	784
10.22.1.64	EXCESS_ON_HAND_AT_END Extension	785
10.23	Active_Strategy Extensions	786
10.23.1	Termination extensions of model Active_Strategy	786
10.23.1.1	NO_PROBLEMS Extension	786
10.23.1.2	TARGET_ACHIEVED Extension	786
10.23.1.3	RESOLVE_COUNT_EXCEEDED Extension	786
10.23.1.4	MANUAL Extension	786

## Contents

10.23.2	execution extensions of model Active_Strategy	786
10.23.2.1	SEQUENCE_RUN_ONCE Extension	786
10.23.2.2	SEQUENCE_RUN_MULTIPLE Extension	787
10.23.2.3	BEFORE_AND_AFTER Extension	787
10.23.2.4	SEQUENTIAL_ALTERNATES Extension	788
10.23.2.5	SEQUENTIAL_ALTERNATES_KEEP_BEST Extension	789
10.23.2.6	PROPORTIONAL_RESOLVES Extension	790
10.23.2.7	ORDERED_RESOLVES Extension	790
10.23.3	problem_selection extensions of model Active_Strategy	790
10.23.3.1	PROPORTIONAL_INTERACTION Extension	790
10.23.3.2	EARLIEST_PROBLEM_START Extension	791
10.23.3.3	SORT_BY_EXPRESSION Extension	791
10.24	Active_Goal Extensions	792
10.24.1	goal_extensions of model Active_Goal	792
10.24.1.1	FEASIBILITY Extension	792
10.24.1.2	MINIMIZE_PROBLEM_COUNT Extension	792
10.24.1.3	MINIMIZE_PROBLEMS Extension	792
10.24.1.4	MINIMIZE_LATENCY Extension	793
10.24.1.5	WEIGHTED_LATENCY Extension	793
10.24.1.6	WEIGHTED_SHORTNESS Extension	793
10.24.1.7	MINIMIZE_SHORTNESS Extension	793
10.24.1.8	MINIMIZE_COST Extension	794
10.24.1.9	MINIMIZE_PROFIT Extension	794
10.24.1.10	MAXIMIZE_REVENUE Extension	794
10.25	Item Extensions	796
10.25.1	spec_extensions of model Item	796
10.25.1.1	STANDARD Extension	796
10.25.1.2	CUSTOM Extension	796
10.26	Skill Extensions	797
10.26.1	selection_extensions of model Skill	797
10.26.1.1	PRIMARY Extension	797
10.26.1.2	PREFER_PRIMARY Extension	797
10.26.1.3	EVEN Extension	797
10.26.1.4	MAX_EFFICIENCY Extension	797
10.27	Skill_Resource Extensions	798
10.27.1	efficiency_extensions of model Skill_Resource	798
10.27.1.1	FIXED Extension	798
10.27.1.2	CALENDAR Extension	798
10.28	Configuration Extensions	799
10.28.1	spec_extensions of model Configuration	799
10.28.1.1	STANDARD Extension	799
10.28.1.2	CUSTOM Extension	799
10.29	Operation_Static Extensions	800
10.29.1	identifier_extensions of model Operation_Static	800
10.29.1.1	EARLIEST Extension	800
10.29.2	static_spec_extensions of model Operation_Static	800
10.29.2.1	STARTED Extension	801
10.29.2.2	COMPLETED Extension	801
10.29.2.3	IN_FRONT Extension	801
10.30	Request Extensions	803

Contents

10.30.1	delivery_policy extensions of model Request	803
10.30.1.1	FIXED Extension	803
10.30.2	delivery_naming extensions of model Request	803
10.30.2.1	NUMBERED Extension	803
10.31	Delivery_Request Extensions	804
10.31.1	promising_policy extensions of model Delivery_Request	804
10.31.1.1	BUCKETED_ALL Extension	804
10.31.1.2	BUCKETED_ASAP Extension	804
10.31.1.3	ON_TIME Extension	804
10.31.1.4	ALL Extension	804
10.31.1.5	ALL_ON_TIME Extension	804
10.31.1.6	ASAP Extension	804
10.31.1.7	ASAP_MONTHLY Extension	805
10.31.1.8	BUCKETED_ALLOCATION Extension	805
10.31.1.9	BUCKETED_ALL_MIN_PRICE Extension	805
10.31.1.10	BUCKETED_MIN_PRICE_ASAP Extension	806
10.31.1.11	SHIP_IN_RATIO Extension	807
10.31.2	fulfillment_policy extensions of model Delivery_Request	807
10.31.2.1	ON_TIME Extension	807
10.31.2.2	FULL_QUANTITIES_OF_ALL_ITEMS Extension	807
10.31.2.3	UNRESTRICTED Extension	807
10.32	Promise Extensions	808
10.32.1	delivery_policy extensions of model Promise	808
10.32.1.1	FIXED Extension	808
10.33	Delivery_Promise Extensions	809
10.33.1	fulfillment_policy extensions of model Delivery_Promise	809
10.33.1.1	ON_TIME Extension	809
10.33.1.2	FULL_QUANTITIES_OF_ALL_ITEMS Extension	809
10.33.1.3	UNRESTRICTED Extension	809
10.34	Horizon Extensions	810
10.34.1	bucket_spec extensions of model Horizon	810
10.34.1.1	ONE Extension	810
10.34.1.2	MONTHS Extension	810
10.34.1.3	WEEKS Extension	810
10.34.1.4	DAYS Extension	810
10.34.1.5	DATES Extension	810
10.35	Delivery_Acceptance Extensions	812
10.35.1	fulfillment_policy extensions of model Delivery_Acceptance	812
10.35.1.1	ON_TIME Extension	812
10.35.1.2	FULL_QUANTITIES_OF_ALL_ITEMS Extension	812
10.35.1.3	UNRESTRICTED Extension	812
10.36	Strategy Extensions	813
10.36.1	termination extensions of model Strategy	813
10.36.1.1	NO_PROBLEMS Extension	813
10.36.1.2	TARGET_ACHIEVED Extension	813
10.36.1.3	RESOLVE_COUNT_EXCEEDED Extension	813
10.36.1.4	MANUAL Extension	813
10.36.2	execution extensions of model Strategy	813
10.36.2.1	SEQUENCE_RUN_ONCE Extension	813
10.36.2.2	SEQUENCE_RUN_MULTIPLE Extension	814

Contents

10.36.2.3	BEFORE_AND_AFTER Extension	814
10.36.2.4	SEQUENTIAL_ALTERNATES Extension	815
10.36.2.5	SEQUENTIAL_ALTERNATES_KEEP_BEST Extension	816
10.36.2.6	PROPORTIONAL_RESOLVES Extension	817
10.36.2.7	ORDERED_RESOLVES Extension	817
10.36.3	problem_selection extensions of model Strategy	818
10.36.3.1	PROPORTIONAL_INTERACTION Extension	818
10.36.3.2	EARLIEST_PROBLEM_START Extension	818
10.36.3.3	SORT_BY_EXPRESSION Extension	818
10.37	Problem_Sel Extensions	820
10.37.1	category extensions of model Problem_Sel	820
10.37.1.1	REQUEST_NOT_PLANNED Extension	820
10.37.1.2	REQUEST_PLANNED_LATE Extension	820
10.37.1.3	REQUEST_PLANNED_EARLY Extension	821
10.37.1.4	REQUEST_PLANNED_SHORT Extension	821
10.37.1.5	REQUEST_PLANNED_EXCESS Extension	822
10.37.1.6	PROMISE_NOT_PLANNED Extension	822
10.37.1.7	PROMISE_PLANNED_LATE Extension	823
10.37.1.8	PROMISE_PLANNED_EARLY Extension	824
10.37.1.9	PROMISE_PLANNED_SHORT Extension	824
10.37.1.10	PROMISE_PLANNED_EXCESS Extension	825
10.37.1.11	ACCEPTANCE_NOT_PLANNED Extension	825
10.37.1.12	ACCEPTANCE_PLANNED_LATE Extension	826
10.37.1.13	ACCEPTANCE_PLANNED_EARLY Extension	827
10.37.1.14	ACCEPTANCE_PLANNED_SHORT Extension	827
10.37.1.15	ACCEPTANCE_PLANNED_EXCESS Extension	828
10.37.1.16	PLANNED_BEFORE_CURRENT Extension	829
10.37.1.17	UNRELEASED Extension	829
10.37.1.18	NEEDS_RELEASE Extension	830
10.37.1.19	INCONSISTENT_OPLAN Extension	830
10.37.1.20	OPERATION Extension	830
10.37.1.21	REQUEST_PROMISED_LATE Extension	831
10.37.1.22	REQUEST_PROMISED_EARLY Extension	831
10.37.1.23	REQUEST_PROMISED_SHORT Extension	832
10.37.1.24	REQUEST_PROMISED_EXCESS Extension	833
10.37.1.25	PROMISE_NOT_OFFERED Extension	834
10.37.1.26	PROMISE_NOT_ACCEPTED Extension	834
10.37.1.27	ACCEPTANCE_INCONSISTENT Extension	834
10.37.1.28	REQUEST_QUEUED Extension	835
10.37.1.29	REQUEST_PLAN Extension	835
10.37.1.30	PROMISE Extension	836
10.37.1.31	REQUEST_PLAN Extension	836
10.37.1.32	PROMISE_PLAN Extension	837
10.37.1.33	REQUEST_PROMISE Extension	837
10.37.1.34	DELIVERY_REQUEST_NOT_COORDINATED Extension	838
10.37.1.35	DELIVERY_PROMISE_NOT_COORDINATED Extension	838
10.37.1.36	DELIVERY_ACCEPTANCE_NOT_COORDINATED Extension	838
10.37.1.37	SUPPLY_PLANNED_LATE Extension	839
10.37.1.38	SUPPLY_PLANNED_EARLY Extension	839
10.37.1.39	SUPPLY_PLANNED_SHORT Extension	840

Contents

10.37.1.40	SUPPLY_PLANNED_EXCESS Extension	840
10.37.1.41	SUPPLY_PROMISED_LATE Extension	841
10.37.1.42	SUPPLY_PROMISED_EARLY Extension	842
10.37.1.43	SUPPLY_PROMISED_SHORT Extension	842
10.37.1.44	SUPPLY_PROMISED_EXCESS Extension	843
10.37.1.45	SUPPLY Extension	843
10.37.1.46	SUPPLY_PLAN Extension	844
10.37.1.47	SUPPLY_PROMISE Extension	844
10.37.1.48	UNIDENTIFIED_OP_STATE Extension	845
10.37.1.49	UNCONSOLIDATED Extension	845
10.37.1.50	UNCOORDINATED Extension	846
10.37.1.51	CONSOLIDATION_OVERSIZE Extension	846
10.37.1.52	CONSOLIDATION_UNDERSIZE Extension	846
10.37.1.53	OVERLOAD Extension	846
10.37.1.54	OVERSIZE Extension	847
10.37.1.55	BUCKET_OVERSIZE Extension	848
10.37.1.56	UNDERLOAD Extension	848
10.37.1.57	RESOURCE Extension	848
10.37.1.58	NEGATIVE_ON_HAND Extension	849
10.37.1.59	OVER_FLOW_LIMIT Extension	849
10.37.1.60	NEGATIVE_ON_HAND_AT_END Extension	850
10.37.1.61	LOT_OVER_CONSUMED Extension	851
10.37.1.62	LOT_NOT_CONSUMED Extension	851
10.37.1.63	LOT_NOT_PRODUCED Extension	851
10.37.1.64	LOT_OVER_PRODUCED Extension	852
10.37.1.65	LOW_ON_HAND Extension	852
10.37.1.66	EXCESS_ON_HAND Extension	853
10.37.1.67	EXCESS_ON_HAND_AT_END Extension	854
10.37.1.68	BUFFER Extension	854
10.38	Strategy_Change Extensions	855
10.38.1	change_category extensions of model Strategy_Change	855
10.38.1.1	MOVE_IN Extension	855
10.38.1.2	MOVE_OUT Extension	855
10.38.1.3	SPLIT Extension	855
10.38.1.4	USE_MORE Extension	855
10.38.1.5	USE_LESS Extension	855
10.38.1.6	USE_ALTERNATE_OPERATION Extension	855
10.38.1.7	USE_ALTERNATE_RESOURCE Extension	855
10.38.1.8	USE_EFFECTIVE_ALTERNATE Extension	856
10.38.1.9	INCREASE_CAPACITY Extension	856
10.38.1.10	DECREASE_CAPACITY Extension	856
10.38.1.11	RESIZE_MORE Extension	856
10.38.1.12	RESIZE_LESS Extension	856
10.38.1.13	UPSTREAM Extension	856
10.38.1.14	DOWNSTREAM Extension	856
10.39	Strategy_Lock Extensions	857
10.39.1	spec extensions of model Strategy_Lock	857
10.39.1.1	OPERATION_PLAN_RANK_RANGE Extension	857
10.39.1.2	OPERATION_PLAN_RANK_EXPRESSION Extension	857
10.40	Strategy_Goal Extensions	858

Contents

10.40.1	goal extensions of model Strategy_Goal	858
10.40.1.1	FEASIBILITY Extension	858
10.40.1.2	MINIMIZE_PROBLEM_COUNT Extension	858
10.40.1.3	MINIMIZE_PROBLEMS Extension	858
10.40.1.4	MINIMIZE_LATENCY Extension	859
10.40.1.5	WEIGHTED_LATENCY Extension	859
10.40.1.6	MINIMIZE_SHORTNESS Extension	860
10.40.1.7	WEIGHTED_SHORTNESS Extension	860
10.40.1.8	MINIMIZE_COST Extension	860
10.40.1.9	MAXIMIZE_PROFIT Extension	860
10.40.1.10	MAXIMIZE_REVENUE Extension	861
10.41	Calendar_Entry Extensions	862
10.41.1	value extensions of model Calendar_Entry	862
10.41.1.1	NUMBER Extension	862
10.41.1.2	QUANTITY Extension	862
10.41.1.3	NUMBER_QUANTITY Extension	862
10.41.1.4	SYMBOL Extension	862
10.41.1.5	TIME Extension	862
10.41.2	day_pattern extensions of model Calendar_Entry	863
10.41.2.1	EVERYDAY Extension	863
10.41.2.2	EVERY_N_DAYS Extension	863
10.41.2.3	WEEKENDS Extension	863
10.41.2.4	DAYS_OF_WEEK Extension	863
10.41.2.5	DAYS_OF_MONTH Extension	864
10.41.2.6	DAY_OF_WEEK_OF_MONTH Extension	864
10.41.2.7	DAY_OF_WEEK_OF_MONTH Extension	865
10.41.2.8	DAY_OF_LAST_WEEK_OF_MONTH Extension	866
10.41.2.9	DAY_OF_YEAR Extension	866
10.41.2.10	YEARLY Extension	867
10.42	Calendar Extensions	868
10.42.1	entry_value extensions of model Calendar	868
10.42.1.1	UNSPECIFIED Extension	868
10.42.1.2	NUMBER Extension	868
10.42.1.3	QUANTITY Extension	868
10.42.1.4	NUMBER_QUANTITY Extension	868
10.42.1.5	SYMBOL Extension	869
10.42.1.6	TIME Extension	869
10.43	Flow_Criterion Extensions	870
10.43.1	criterion extensions of model Flow_Criterion	870
10.43.1.1	CUSTOMER_RANK Extension	870
10.43.1.2	SELLER_RANK Extension	870
10.43.1.3	REQUEST_RANK Extension	870
10.43.1.4	ACTUAL_OR_FORECAST Extension	871
10.43.1.5	PROMISED_EXTENSION Extension	871
10.43.1.6	PROMISED_EXTENSION Extension	872
10.43.1.7	REQUEST_DUE Extension	872
10.43.1.8	DUE_DATE Extension	873
10.43.1.9	PROMISED_EXTENSION Extension	873
10.43.1.10	ENTRY_DATE Extension	874
10.44	Calendar_Plan Extensions	875

## Contents

10.44.1	entry_value extensions of model Calendar_Plan .....	875
10.44.1.1	NUMBER Extension .....	875
10.44.1.2	QUANTITY Extension .....	875
10.44.1.3	NUMBER_QUANTITY Extension .....	876
10.44.1.4	SYMBOL Extension .....	877
10.44.1.5	TIME Extension .....	878
10.45	Format Extensions .....	879
10.45.1	spec extensions of model Format .....	879
10.45.1.1	Void Extension .....	879
10.45.1.2	Logical Extension .....	879
10.45.1.3	String Extension .....	880
10.45.1.4	Symbol Extension .....	880
10.45.1.5	Date Extension .....	881
10.45.1.6	Date_Range Extension .....	883
10.45.1.7	Number Extension .....	883
10.45.1.8	Percentage Extension .....	885
10.45.1.9	Integer Extension .....	886
10.45.1.10	Quantity Extension .....	889
10.45.1.11	Quantity_Range Extension .....	890
10.45.1.12	Time Extension .....	891
10.45.1.13	Restuction Extension .....	892
10.45.1.14	Horizon_Date Extension .....	893
10.45.1.15	List Extension .....	895
10.46	Field Extensions .....	897
10.46.1	field_type extensions of model Field .....	897
10.46.1.1	SIMPLE Extension .....	897
10.46.1.2	SELECTOR Extension .....	897
10.46.1.3	EXTENDED Extension .....	897
10.46.1.4	USER Extension .....	897

# 1 Introduction

## 1.1 Preface

You may wish to skip Chapter 1, Chapter 2 (which introduces the basic types) and Chapter 3 (which discusses basic worksheet functions), and jump immediately into the Supply Chain model in Chapter 4. You can return to the basic types and functions as you encounter them in the models. You may need to return to this Introduction for a quick overview of OIL Expressions if you find examples to be unclear, or for a quick overview of the other manuals available to you.

However, most will benefit from at least reading this Introduction, even if you choose to skip the basic types and functions in chapters 2 and 3.

## 1.2 Overview

RHYTHM Supply Chain Planner (SCP) is an innovative decision support system designed to give supply chain decision makers unprecedented visibility of their supply chain and intelligent suggestions on how to better plan it. It is designed to work cooperatively with all the decision makers in the supply chain (the planners, managers, sales personnel, etc.) to both be more effective in their domain and to be more effective with the other decision makers involved. As such it must be customizable to the needs of each of the varied decision makers in the supply chain. SCP is designed to be easy to customize by its users (planners, managers, salespeople -- not programmers).

SCP is a Truly Integrated Planning (TIP) system. There are many books, papers, and advertising on the importance of "integrated planning". However, the "integrated" solutions that are presented are consistently dis-integrated. The solutions either involve getting dis-integrated planning packages to communicate, or bringing dis-integrated planning tools into one package. Putting one user interface over a set of tools and one database under a set of tools does not integrate the planning performed by the independent tools. And it does not provide the decision makers with the visibility and capabilities needed to perform Truly Integrated Planning.

Truly Integrated Planning involves integration of the material and capacity planning, integration of factory and distribution planning, integration of master and operational (execution) planning, integration of manual and automated planning, integration of make-to-stock and make-to-order planning, integration of discrete and repetitive planning, and more.

Effective supply chain management depends upon the ability to have integrated plans throughout the supply chain. Supply chains are typically diverse in their needs. Repetitive suppliers

will feed discrete manufacturers. An essentially make-to-stock facility will be fed by a make-to-order facility and will feed standard components to an assemble-to-order packaging plant. The key to success is at the customer interface, so no matter how well the manufacturing facility performs, if the distribution plan is faulty the manufacturer will suffer. If the key to success is in satisfying their customers, the decision makers must have visibility of the complete supply chain responsible for that satisfaction in order to make the decisions that will optimize that satisfaction.

For a more thorough discussion of Truly Integrated Planning and the other concepts on which SCP is based, see the *RHYTHM Supply Chain Planner Concept Manual*.

### 1.3 RHYTHM Manuals

The *RHYTHM SCP Concept Manual* describes the basic concepts on which SCP is based. It discusses Truly Integrated Planning (TIP), advanced supply chain management with TIP, customization to support TIP, and many of the other innovative features that TIP demands and RHYTHM SCP provides.

This manual, the *RHYTHM SCP Model Reference Manual*, describes the models that the user can set up and access via the Interactive Reports. Each model consists of fields, and the fields are functions in the Report's spreadsheet-like expression language. This information is available through online help.

The models and fields define all the functionality that can be accessed by the Interactive Reports, and thus almost all the functionality available from the RHYTHM product suite.

The *RHYTHM SCP Customization Manual* provides the tools and information necessary to perform the tasks involved in customizing RHYTHM SCP. The manual consists of reference materials designed to supplement the RHYTHM SCP Model Reference Manual. This manual is also available as online help.

The *RHYTHM SCP User Manual* provides users with basic conceptual information, how to information, and detailed information for each of the workbenches, menus, and windows within the interface. This manual is only available as online help and is accessed from the Help Topics option of the Help menu.

The *RHYTHM SCP Q&A Manual* is designed around "How do I use SCP to do X?" questions. It is designed to cover commonly asked questions. It is designed to map common scenarios with other paradigms to SCP. It is designed to present challenging modeling or usage problems and illustrate one or more solutions. It is designed to help show the flexibility and power of the mechanisms by presenting a variety of problems and showing a variety of solutions.

## 1.4 Modeling

### 1.4.1 Expressions

Expressions, the Object Interaction Language, and all of their applications in the Interactive Reports are discussed in detail in the *RHYTHM SCP Customization Manual*. However, to keep this manual stand-alone, a brief introduction to Expressions is appropriate.

OIL Expressions are very much like traditional spreadsheet expressions. An expression evaluates to a data value. There are various functions which can take one or more data values as parameters and compute a new data value.

### 1.4.2 Types

Each data value in a model has a data "type". For instance, two common types are 'Number' and 'String'. The functions take parameters with specific types and evaluate to a specific type of data value. Formatting of data for display is type-specific. Controls for interacting with and editing data values are type-specific. Help is type-specific. And so on.

There are a number of basic types, such as 'Number', 'Quantity', 'Date', 'Time', and 'String'. All the basic types are described in chapter 2.

There is also a List type. The List type holds any number of elements, where each of the elements is a data value of the same type. The type of the data elements in the List is specified within brackets [], such as List[Number] or List[Date].

Beyond the typical data types, each model described in this manual is a "model type" in the Object Interaction Language. For example, 'Operation', 'Resource', 'Buffer', 'Product', 'Site', 'Supply\_Chain', and 'Plan' are all types, specifically, model types. The term "model" is used to mean an instance/value of a "model type" (for object-oriented programmers, this equates nicely to "objects" which are instances of a "class"). For example, the DRILL32 may be a model of the Resource model type. So, an Expression can compute a Resource, DRILL32, which may be passed as a parameter to another function in the Expression to compute Calendar for that Resource.

Finally, Expressions are themselves a data type in the Object Interaction Language. Thus, Expressions can evaluate to other Expressions; and fields of models can hold Expressions which are evaluated by the planning engine to compute values dynamically. That provides programmability in a form that is identical to writing Interactive Reports, and thus familiar to most TIPS users. (In fact it is familiar to most spreadsheet users.)

### 1.4.3 Fields

In addition to the functions typically found in traditional spreadsheets, each of the fields in this manual are functions in the Object Interaction Language. In fact, that is the purpose of the



Object Interaction Language: to give flexible access to the data and functionality in the models that are being created, edited, and planned.

The typical field is a function that takes a model (a data value of a model type) as an argument and returns a particular data value out of that model. For example, the 'name' field function will take a Resource as an argument and will return the Symbol that is the name or id of that Resource; the 'location' field function will take a Resource and return a Location which models the location of that Resource.

Fields can take additional arguments. For example, the 'load\_time' field takes a Resource\_Plan and a Date\_Range. It returns the hours (Time) of load that is planned on that Resource during that Date\_Range.

Some fields are view-only. That is, they are not settable by the user. They are analysis outputs, not user inputs. View-Only is just one of the possible properties of a field which are documented after the field description.

All the modeling and planning functionality provided by SCP is provided through field functions described in this manual. Almost all windows and reports available in SCP were created with Expressions and constructs of the Object Interaction Language, using field functions that can be found in the *RHYTHM SCP Model Reference Manual*. Thus, most anything you see in SCP, you can customize or wholly rewrite.

#### 1.4.4 Models

This manual describes the models that can be used to model your supply chain, its current state, and plans for it. For example, there are models for Operations, Resources, Buffers, Products, Sites, Supply\_Chains, Plans, and Problems.

A model is essentially defined by the fields that make it up. A field defines a data value that describes a characteristic of the real-world thing being modeled. For example, the location of a machine: the machine is modeled as a Resource, the location is modeled as a Location, and the Resource has a field 'location' that specifies the Location of that Resource.

#### 1.4.5 Submodels

The SCP models form a hierarchy. For example, a Supply\_Chain model contains any number of Site models, Seller models, and Plan models. Site models contain Operation, Resource, and Buffer models. Operation is a submodel of Site, which is a submodel of Supply\_Chain.

Each submodel has an 'owner' field which returns the model that it is a submodel of. And that owning model has a field for the submodels which is a List of that submodel type containing each submodel owned by it.

For example, Resource's 'owner' field returns the Site that owns that Resource. And Site has a field 'resources' which contains all the Resource models that it owns.

Many models have a key field, a unique identifier for that model within its owner. For instance, Resource's key field is 'name'. Each Resource in a Site must have a unique name. And given a Site and a name, it is possible to find the Resource model with that name.

Except for the key field, all fields of a model will default to some meaningful value if a value is not supplied. The default values for each field are documented after the field description. This defaulting eases creation of new models.

#### 1.4.6 Extensions

The SCP models are based upon our Extensible Model Architecture™, which is a unique approach to modeling that allows a model to be extended with additional fields depending upon the value of a particular field.

In this way, a simple model can be provided to support the majority of scenarios. But for the few instances that require much more modeling sophistication, the model can be extended with additional information and additional semantics, without imposing cost on the rest of the models.

Similarly, a single product (RHYTHM SCP) can provide all of the functionality needed to model the various elements that make up a supply chain without burdening the models of users that do not need all that different functionality. In that way, different users only have to deal with what they want to deal with.

A field identified as an extension selector specifies which extension to use for a particular semantic. Extensions define the semantic behavior of the model when that extension is added, and the additional fields that are made available on that model.

#### 1.4.7 User-Defined Fields

Some model types can be extended by the addition of user-defined fields, which can be of any data type specified by the user. Such user-defined fields are available for use as functions in expressions as if they were built-in fields. The planning algorithms can even be parameterized such that they read the user-defined fields in performing planning actions.

Not all model types can be extended in this way. Some model types are very light-weight components of other models. As such, allowing user-defined fields on each of the components would be quite expensive (computationally). Some models are transient in nature. Allowing user-defined fields would be somewhat deceiving since they would disappear at the whim of the algorithms.

## 1.5 Manual Organization

### 1.5.1 HyperText

This is a hypertext manual. Hyperlinks are printed in bold. Most type names are links, most model names are links, most field functions are links, and so on. You can utilize these links by viewing with FrameMaker, Frame Viewer, or an HTML (WWW) Browser (such as Mosaic).

### 1.5.2 Notation

Type names (and thus model type names) are written capitalized (e.g., Supply\_Chain). Field function names are written in single quotes, as are Expressions (e.g., 'sites' or 'A1 + B2'). Data values are written in double quotes (e.g., "36 gal"). Extension names are typically in all capital letters (e.g., PRODUCE\_YIELD\_CALENDAR).

### 1.5.3 Chapters

The next chapter discusses all the Basic Types. The third chapter defines the Basic Functions, including many of those that a spreadsheet user would find familiar. The fourth chapter defines all the models. The fifth chapter provides the catalog of extensions that can be chosen. An index follows.

### 1.5.4 Fields

Each field is documented with a consistent, stand-alone block, which is suitable as well for on-line Help display. The redundant information from a manual point-of-view is in a standard header line that states the type of the field; whether it is a field, submodel, or extension; and the model that contains it.

The description of that field follows. After the description, the default value is named and then if there are any properties (such as View-Only), they are documented.

Extension selector fields will each be followed by a list of all the extensions, providing a mini-index to those extension descriptions.

### 1.5.5 Models

The models are documented in chapters, hierarchically. Submodels are documented in sub-chapters of their 'owner' model.

The model header gives the name and the name of the 'owner' model. That is followed by the description of that model and how it is used to model real world things.

Various summaries are then provided, such as what submodels it contains, and what extensions. Since these are links, it provides mini-indices per model chapter.

That is followed by each of the field descriptions as described previously.

Then in subchapters come each of the submodels of that model. Also in subchapters is the submodels of extensions.

### 1.5.6 Extensions

The extensions are documented in the Extensions chapter, which is organized with a subchapter per model, and a sub-subchapter per extension selector of that model.

Each extension header gives its name, the model that it is an extension of, and the extension selector used to select it. That is followed by a description of the semantics that the extension adds to its model. Finally, descriptions of the fields that are added to the model by that extension are described, as discussed previously.

---

## *Introduction*

---

## 2 Basic Types

### 2.1 Cell\_State

Cell State

### 2.2 Computed\_String

This is an obsolete type that will soon disappear. All uses of Computed\_String will become simply String.

For now, it distinguishes a "computed" String, one that is not just a view of a String that lives elsewhere, but rather actually holds the characters.

### 2.3 Date

A particular point in history (past or future), with precision of one second. For example, the Date "1989 Dec 19 12:50:32". A Date must be in the range 1970 to 2099. A Time can be added to a Date, producing a Date that much Time in the future. Two Dates can be subtracted resulting in a Time that is equal to the duration between those two Dates.

The Dates "infinite\_past" ("-----") and "infinite\_future" ("+++++") are also supported, with normal infinite semantics (adding to or subtracting from an infinite Date, results in the same infinite Date; subtracting two infinite Dates is an error).

Note that in some systems the "date" fields just model the day information, and separate "time-of-day" fields give the time within that day. This is simple for formatting and purely database-ish applications. However, such separation is illogical and cumbersome for systems that allow sophisticated computations on the Dates. Thus, a Date models all that is needed to "make a date with someone", both the day and the time-of-day. And Time models duration (e.g., number of hours).

All times are stored internally as seconds from some particular time in the past. So, there is no hard coding of centuries, etc. Sorting is done before converting the date to text, so the sequence will always be correct, even if the date format does not include the year at all. The sort before converting to text works for quantities, too. For example, "1 yr" is bigger than "20 day". Specific conditions that are satisfied include:

- \* 20th century dates in DD/MM/YY format entered retrospectively once the year 2000 is reached are interpreted correctly as 20th century dates, e.g. 12/05/98.
- \* Dates in DD/MM/YY format entered in the 20th century for future dates in the 21st century are interpreted correctly as being the 21st century dates, e.g. 01/03/01.

\* Data queries that cross the millennium, e.g. between 1998 and 2002 return the correct data, instead of an error or no data.

When parsing any data format which has a 2 digit year, years less than 70 are assumed to be in the 21st century. For example, 01-01-01 with format YY-MM-DD is January 1, 2001.

### 2.4 Date\_Range

A range of Dates from a 'start' Date up to (but not including) an 'end' Date. Functions 'start' and 'end' can get and set the bounds of a Date\_Range. Beyond being just a pair of Dates, a Date\_Range supports special formatting. For example, a Date\_Range can be displayed or input as "July 1991", "Week 12 1993", "3Q 1995", or "95-02-12 / 95-04-15".

A Date\_Range can be created from two Dates (date\_range(date1, date2)) or from a start Date and a Time (date\_range(start, time)).

There are also special functions to generate lists of consecutive Date\_Ranges, which is perhaps the primary purpose for the Date\_Range type. For example, months(start, end) returns a List of Date\_Ranges (List(Date\_Range)) that start and end on month boundaries. Similarly, 'days', 'weeks', 'quarters', and 'years' are available. Combinations of those can be formed by using the 'list' function to combine lists from different of those functions. For example, list(weeks(start, start+"4 weeks"), months(start+"4 weeks", end)) provides 4 weeks followed by several months.

### 2.5 Event

A User interaction or other occurrence that should be "handled" by a GUI client. For example, a key press or release, or a mouse-button press, drag, or release.

### 2.6 Expression

The value is a expression that can be evaluated to compute something. The expression syntax is the same as is available in the Report Worksheet.

### 2.7 Func

This type is obsolete.

The value is a function which can be passed into other functions, called and executed. It is used in particular by List\_for\_each(Function), which executes the Function for each element of the List.

2.8 Horizon\_Date

Horizon\_Date is a Date specified relative to the start of the planning horizon. Thus, as time passes, and the planning horizon moves forward, so do these Dates. Horizon\_Date is used primarily to specify time fences, which are necessarily horizon-relative.

It allows fairly sophisticated specification of relative dates, such as the second Monday, the first day of the third month, the fourth Thursday of the second month, and so on.

It has minute granularity: so you can specify a Date at 8:30am, but you cannot specify a Date at 8:30:22am.

Currently, OIL treats the time part of the horizon date not as an offset, but as a specific time of day. So, if the horizon date is listed as '1 day at 11:00', the resulting horizon will be the next day at 11:00. Because time is absolute and not relative, you can not have a horizon\_date that is only a time. The time component of a horizon date is optional. If not specified, a time of 00:00 is assumed.

2.9 ID

Used by charts to send unique identifiers to the GUI. There's an automatic conversion from any model-type to ID.

2.10 Integer

An integer (number with no fractional part) in the range from 2 billion down to -8 million (approximately). This is typically used for small whole numbers that specify the number of things, or which one of that number of things (e.g., the 15th of 240). Using Numbers for that allows absurdity (e.g., 15.74th of 240.381).

2.11 Logical

A truth value: either true or false.

2.12 Measure

A physical measure or dimension. For example, mass, length, time, money, temperature, and piece\_count are all base Measures. Measures can be composed from the base measures. For example, volume is length[3], area is length[2], velocity is length\*time[-1], force is mass\*length\*time[-2], and money\*time[-1], mass\*time[-1], and so on are all possible.

There are 8 base Measures. Time and Money are predefined as immutable. The other 6 can be largely redefined as most useful for the manufacturer. For instance, temperature may have no meaningful usage for a manufacturer, so he can effectively use it to measure in "number of cans".

2.13 Measure\_Unit

A Measure\_Unit is a Measure plus preferred units in which to display a Quantity with that Measure. It is used primarily to specify how to convert Quantities. A Quantity may be converted to use different units of the same Measure; or may be converted to a different unit and Measure of a particular Item, Operation, etc.

For example, mass, length, volume, time, velocity, piece\_count, and rate are all Measures. A Quantity with Measure mass can be displayed or input in various units; for example, "kg", "g", "pounds", or "tons". Similarly, length could be in "km", "cm", "miles", or "ft"; time could be in "s", "min", "hr", "days", or "weeks"; velocity could be in "km/hr" or "ft/s". And so on.

One unit of a particular Item may be defined to be "100 kg", "50 liters", and "\$750". Thus, a unit-of-an-Item is defined in terms of one or more unit-of-Measure. A Quantity of "400 kg" of that Item may then be converted to Measure\_Unit "\$" (which would be "\$3000") or to Measure\_Unit "ml" (which would be "200000 ml").

The base units which can be used by Measure\_Unit for a particular Measure are defined by the Measure\_Base models. Measure\_Base defines that "hr" is "60 min", and that "min" is "60 s". Measure\_Base models can be defined by the user.

Note that Measure\_Base defines a conversion from one unit of a Measure to another unit of the same Measure. Such Measures, units, and conversions apply to anything. The Unit model defines a conversions between units of different Measures that are specific to an Item, Buffer, Operation, or Product. Such Units and conversions are not generic like units of Measures.

2.14 Money

Money is effectively a special case of Quantity. It is a Quantity with Measure 'money', a currency value such as "\$13" or "4800 yen". It is separated out because of its importance in representing costs, and its special needs for "native" currency at different Sites. It can be easily intermixed in expressions with Quantity's, but where Money is called for, the Measure of the Quantity must be simply 'money'.

Note that Money formatting can be different than normal Quantities.

2.15 Number

A real number (can have fractional part) in the range from 1e37 (1 with 37 0's) down to -1e37 (approximately). It can represent numbers as small in magnitude as 1e-45 (decimal point, 45 0's, 1).

However, it has only limited precision: approximately 6 decimal digits.

So,  $100,000,000 + 100 = 100,000,096$

$100,000,000 + 1 = 100,000,000$  (no change) Another example: typing 123456789 will result in 123456792 (only the first 7 digits are accurate). Luckily, there are very few applications that require more than 6 digits of precision.

2.16 Pathname

Specifies the name of a file or directory in the host's file system.

2.17 Percentage

Percentage is simply a Number. Mathematically it is used just like any Number. It is formatted the same as any Number. It freely converts to a Number, and Numbers can be freely converted to Percentages.

The only difference is that the value is typically displayed in percentage form ("50%") rather than normal form ("0.5"). By defining such fields to have a Percentage type rather than Number type, the user can set up different default Formats for all the Percentage fields.

Note that inputting "1" is equivalent to "100%", and "100" is equivalent to "10000%".

2.18 Predicate

A variable with type Predicate specifies a predicate descriptor.

2.19 Priority

A function execution priority.

2.20 Property\_List

This is an internal type and should not be visible to users.

A list of properties is used to allow users to extend models by adding their own fields.

2.21 Quantity

An amount of something, specified by a Number and a Measure\_Unit. For instance, "32 hr", "15 ft", "22 kg", "\$3620", and "16" are all Quantities. The last is a unitless Quantity, which is freely convertible to a simple Number.

The 'convert' functions allow a Quantity to be converted from one Measure\_Unit to another (the Measures must be the same). For instance, convert("kg", "4.4 lbs") will result in "2 kg". Quantities that are combined together will tend to combine or merge the Measure\_Units into the result. For instance, "33 kg" / "3 hr" will result in "11 kg/hr".

Note that "0" of any Measure can be converted to "0" of any other. Also, convert will invert a number to get a match. For instance, "10 unit/hr" can be converted to "hr/unit". It will be "0.1 hr/unit". Or it can be converted to "min/unit", which will give "6 min/unit". This ability to invert on conversions can greatly ease use of rates.

2.22 Quantity\_Range

A range of Quantity from a 'min' Quantity up to a 'max' Quantity, inclusive. Functions 'min' and 'max' can get and set the bounds of a Quantity\_Range. Beyond being just a pair of Quantity's, a Quantity\_Range supports special formatting. For example, a Quantity\_Range can be displayed or input as "[2.5, 7.5]", "2.5 + 5", "5 + 2.5", or "2.5 to 7.5".

A Quantity\_Range can be created from two Quantity's 'quantity\_range(q1, q2)';

2.23 Record

A variable with type Record specifies a RhythmLink data record.

2.24 Reference

An indirect pointer to a value which may be in a worksheet cell or a variable. There is an automatic conversion from 'Cell\_State' to 'Reference'.

2.25 Restriction

A Restriction restricts a Date or Date\_Range to a certain set of values. It can restrict either the "start" or the "end" of a Date\_Range, or the whole Date\_Range. It can specify a Date\_Range that the Date(s) must be "within" or "not within". Note that "start after <some Date>" actually functions as though specifying a start within date and a date in the infinite future". This means that the restriction restricts the date to a start time within the range between some date following the date specified and a date in the future. Also, "end before <some Date>" functions as though specifying an end time within the infinite past and the specified date. This means that the restriction restricts the end date to a range between some date in the future and the date specified. To specify infinite future, type '+' or '.'. For example, setting the restriction to "start first in 97-10-15 00:00 / +." will immediately change this hint to "start after 97-10-15 00:00" in the display. An empty Restriction "" will cause an error. However, it is possible to specify [], which automatically changes to [unspecified] (the default value).

Restrictions are commonly used to temporarily guide the automated planning process (a hint). Such hints can be imposed by manual intervention. The engine then tries to follow it, and then the hint is done. The hint is no longer followed during any subsequent planning or strategy runs.

Note this is specifically a Restriction of a Date or Date\_Range. There are many other hints and locks that "restrict" the plan in other ways, but those are not of type Restriction.

2.26 Safety\_Stock\_Units

An enumeration value used by the CALENDAR stocking\_policy extension of the Buffer model, in the default\_user\_bias\_type field. See that field for details.

2.27 String

A text string: simply a sequence of letters, numbers, punctuation, spaces, tabs, line-feeds, etc. It can be of any length.

Most types can be converted (or formatted) to and from a String.

2.28 Symbol

A String that is used as the name or identifier of a model or object. It can be freely converted to a String with no change, and used wherever a String can. A String can be freely converted to a Symbol, but leading and trailing whitespace (spaces, tabs, line breaks, etc.) will be trimmed off (leaving such characters on identifiers can lead to data errors that are hard to see and correct).

2.29 Time

Time is effectively a special case of Quantity. It is a Quantity with Measure 'time', a duration such as "13 hr". It is separated out because of its importance, allowing it to be handled specially, and restricted appropriately (Times cannot be assigned Quantities such as "15ft"). It can be easily intermixed in expressions with Quantity's, but where a Time is called for, the Measure of the Quantity must be simply 'time'.

Note that Time formatting can be different than normal Quantities. In particular, the format "hh:mm:ss" is supported for Times. This is particularly useful when the Time field represents the time since the start of a day. Thus, adding the Date "91-07-23" to the Time "12:52:32" results in the Date "91-07-23 12:52:32", as would be expected.

Also note that in some systems the "time" models the time-of-day portion of a Date, and "date" models only the day information. This is simple for formatting and purely database-ish applications. However, such separation is illogical and cumbersome for systems that allow sophisticated computations on the Dates. Thus, a Date models all that is needed to "make a date with someone", both the day and the time-of-day. And Time models duration (e.g., number of hours). However, as noted above, where a Time field means "since the start of a day", a Time field can seem very much like other systems "time-of-day" fields: 12:52 means 12 hours and 52 minutes since the start of the day, but it also looks like a time-of-day in 24-hour format.

2.30 Type

A variable with type Type specifies some type, such as Quantity, Date, Symbol, or even Type itself.

2.31 Typed\_Value

A pair, comprising a value and the type of the value. A Typed\_Value may specify, for example, a model instance and the Model\_Type\* which describes it.

2.32 Void

No value is expected or returned.

3 Basic Functions

3.1 Operators

Table 1: Operators

Operator	Precedence	Parameters	Return Type	Description	Example
-	4	(Integer)	Integer	Integer negate	-9000
-	4	(Number)	Number	Floating negate	-10.2
-	4	(Quantity)	Quantity	Quantity negate	-52
-	4	(Time)	Time	Time negate	-021530
-	4	(Date, Date)	Time	Subtract two dates to get the time difference	11MAY1995000000 - 01MAY1995000000
-	4	(Date, Time)	Date	Subtract a time from a date	11MAY1995000000 - 021530 returns 10MAY1995214430
-	4	(Integer, Integer)	Integer	Subtract two integers	9000 - 1000 returns 8000
-	4	(Number, Number)	Number	Subtract two floating numbers	10.2 - 6.4 returns 3.8
-	4	(Quantity, Quantity)	Quantity	Subtract two quantities	52 - 36 returns 16
-	4	(Time, Time)	Time	Subtract two times to get the time difference	021530 - 010000 returns 011530
-	4	(Date, Time)	Date	Add a time to a date to get a new date	11MAY1995000000 + 021530 returns 11MAY1995021530
-	4	(Time, Date)	Date	Add a date to a time to get a new date	021530 + 11MAY1995000000 returns 11MAY1995021530
-	4	(Integer, Integer)	Integer	Add two integers	9000 + 1000 returns 10000
-	4	(Number, Number)	Number	Add two floating numbers	10.2 + 6.4 returns 16.6
-	4	(Quantity, Quantity)	Quantity	Add two quantities	52 + 36 returns 88
-	4	(Time, Time)	Time	Add two times	021530 + 010000 returns 031530
-	5	(Integer, Integer)	Integer	Multiply two integers	9000 * 1000 returns 9000000
-	5	(Number, Number)	Number	Multiply two floating numbers	10.2 * 6.4 returns 65.28

Table 1: Operators

Operator	Precedence	Parameters	Return Type	Description	Example
*	5	(Quantity, Quantity)	Quantity	Multiply two quantities	52 * 36 returns 1872
/	5	(Integer, Integer)	Integer	Divide two integers	9000 / 1000 returns 9
/	5	(Number, Number)	Number	Divide two floating numbers	10.2 / 6.4 returns 1.594
/	5	(Quantity, Quantity)	Quantity	Divide two quantities	52 / 36 returns 1.44
<	3	(Date, Date)	Logical	Less than date	11MAY1995000000 < 21MAY1995000000 returns TRUE
<	3	(Integer, Integer)	Logical	Less than integer	9000 < 1000 returns FALSE
<	3	(Number, Number)	Logical	Less than floating	10.2 < 6.4 returns FALSE
<	3	(Quantity, Quantity)	Logical	Less than quantity	52 < 36 returns FALSE
<	3	(String, String)	Logical	Less than string (case insensitive)	"ORDER_12" < "ORDER_22" returns TRUE
<	3	(Symbol, Symbol)	Logical	Less than symbol (case insensitive)	"ORDER_12" < "ORDER_22" returns TRUE
<=	3	(Time, Time)	Logical	Less than time	021530 < 010000 returns FALSE
<=	3	(Date, Date)	Logical	Less than or equal date	11MAY1995000000 <= 21MAY1995000000 returns TRUE
<=	3	(Integer, Integer)	Logical	Less than or equal integer	9000 <= 1000 returns FALSE
<=	3	(Number, Number)	Logical	Less than or equal floating	10.2 <= 6.4 returns FALSE
<=	3	(Quantity, Quantity)	Logical	Less than or equal quantity	52 <= 36 returns FALSE
<=	3	(String, String)	Logical	Less than or equal string (case insensitive)	"ORDER_12" <= "ORDER_22" returns TRUE
<=	3	(Symbol, Symbol)	Logical	Less than or equal symbol (case insensitive)	"ORDER_12" <= "ORDER_22" returns TRUE
<=	3	(Time, Time)	Logical	Less than or equal time	021530 <= 010000 returns FALSE
>	3	(Date, Date)	Logical	Greater than date	11MAY1995000000 > 21MAY1995000000 returns FALSE



Table 1: Operators

Operator	Precedence	Parameters	Return Type	Description	Example
>	3	(Integer, Integer)	Logical	Greater than integer	9000 > 1000 returns TRUE
>	3	(Number, Number)	Logical	Greater than floating	10.2 > 6.4 returns TRUE
>	3	(Quantity, Quantity)	Logical	Greater than quantity	52 > 36 returns TRUE
>	3	(String, String)	Logical	Greater than string (case insensitive)	"ORDER_12" > "ORDER_22" returns FALSE
>	3	(Symbol, Symbol)	Logical	Greater than symbol (case insensitive)	"ORDER_12" > "ORDER_22" returns FALSE
>	3	(Time, Time)	Logical	Greater than time	021530 > 010000 returns TRUE
>=	3	(Date, Date)	Logical	Greater than or equal date	11MAY1995000000 >= 21MAY1995000000 returns FALSE
>=	3	(Integer, Integer)	Logical	Greater than or equal integer	9000 >= 1000 returns TRUE
>=	3	(Number, Number)	Logical	Greater than or equal floating	10.2 >= 6.4 returns TRUE
>=	3	(Quantity, Quantity)	Logical	Greater than or equal quantity	52 >= 36 returns TRUE
>=	3	(String, String)	Logical	Greater than or equal string (case insensitive)	"ORDER_12" >= "ORDER_22" returns FALSE
>=	3	(Symbol, Symbol)	Logical	Greater than or equal symbol (case insensitive)	"ORDER_12" >= "ORDER_22" returns FALSE
>=	3	(Time, Time)	Logical	Greater than or equal time	021530 >= 010000 returns TRUE
=	3	(Date, Date)	Logical	Equality for date	11MAY1995000000 = 21MAY1995000000 returns FALSE
=	3	(Integer, Integer)	Logical	Equality for integer	9000 = 1000 returns FALSE
=	3	(Number, Number)	Logical	Equality for floating	10.2 = 6.4 returns TRUE
=	3	(Quantity, Quantity)	Logical	Equality for quantity	52 = 36 returns FALSE
=	3	(String, String)	Logical	Equality for string (case insensitive)	"ORDER_12" = "ORDER_22" returns FALSE

Table 1: Operators

Operator	Precedence	Parameters	Return Type	Description	Example
==	3	(Symbol, Symbol)	Logical	Equality for symbol (case insensitive)	"ORDER_12" == "ORDER_22" returns FALSE
==	3	(Time, Time)	Logical	Equality for time	021530 == 010000 returns FALSE
!=	3	(Date, Date)	Logical	Inequality for date	11MAY1995000000 != 21MAY1995000000 returns TRUE
!=	3	(Integer, Integer)	Logical	Inequality for integer	9000 != 1000 returns TRUE
!=	3	(Number, Number)	Logical	Inequality for floating	10.2 != 6.4 returns TRUE
!=	3	(Quantity, Quantity)	Logical	Inequality for quantity	52 != 36 returns TRUE
!=	3	(String, String)	Logical	Inequality for string (case insensitive)	"SAW" != "SAW_SAND" returns TRUE
!=	3	(Symbol, Symbol)	Logical	Inequality for symbol (case insensitive)	"SAW" != "SAW_SAND" returns TRUE
!=	3	(Time, Time)	Logical	Inequality for time	021530 != 010000 returns TRUE
%	5	(Integer, Integer)	Integer	Integer remainder	20 % 6 returns 2 (20 / 6 = 3 with a remainder of 2)
%	5	(Number, Number)	Number	Floating remainder	
&	7	(String, String)	String	String concatenation	"ORDER_1" & ".MFG0001" returns "ORDER_1.MFG0001"
and	2	(Logical, Logical)	Logical	Logical AND	If A is TRUE and B is TRUE, then A and B returns TRUE
or	2	(Logical, Logical)	Logical	Logical OR	If A is TRUE and B is FALSE, then A or B returns TRUE
!	2	(Logical)	Logical	Logical NOT	If A is TRUE, then !A returns FALSE

### 3.2 List Functions

Lists are sequential collections of elements of the same kind.

Here are some examples of lists:

{ "hello", "world" } - a list of two strings  
{ 1, 2, 3, 4 } - a list of four integers  
{ a, b, c, d, e } - a list of five symbols

It is not possible to mix different kinds of elements into a list:  
(1, a, "hello") - illegal list; it must only contains elements of the same kind.

The elements within a list do not need to be unique:  
{1, 2, 3, 2, 1} - a list of five integers, with elements 1 and 2 repeated twice.

Lists can be created, queried, filtered, sorted, and manipulated via the functions defined below.

3.2.1 count ( List[Void] )  
count determines how many elements are in a List.

Returns: the number of elements in the List.

Parameters: - a List of any type

Example Usage:  
count(list(5, 10, 15)); // returns 3  
list("a", "b", "c", "d").count; // returns 4

3.2.2 filter ( List[Void], Expression, Integer )  
filter traverses a list copying those elements which match an expression into a new filtered list. The expression can contain the special character '#' which filter substitutes with each element in the list.

Returns: a List of filtered elements.

Parameters: - a List of any type  
- an expression which must return Logical  
- an optional count of the maximum size for the result. The default is infinite.  
Note: filter(list, expr).first automatically sets the last parameter of the filter function to one.

Example Usage:  
list(2, 4, 6, 8).filter(# < 5); // returns list(2, 4)  
list(2, 4, 6, 8).filter(# < 5, 1); // returns list(2)

filter(list(1, 2, 3), # > 5); // returns an empty list

3.2.3 for\_each ( List[Void], Expression )  
for\_each traverses a list applying an expression to each element and placing the result in a new List. The expression can contain the special character '#' which filter substitutes with each element in the list.

Returns: a result List with the same number of elements as the given list.  
Its elements will have the same type as the expression's return type.

Parameters: - a List of any type  
- an expression

Example Usage:  
list(8, 6, 4, 2).for\_each(# / 2); // returns list(4, 3, 2, 1)  
list(1, 2, 3).for\_each(#).sum; // returns 6

3.2.4 sort ( List[Void], Expression )  
sort orders a list based on an expression. sort processes a list until the sorting expression is true for all adjacent elements. Use 'a' and 'b' as element place holders in the expression.

sort uses a fast, non-stable quicksort algorithm. Because it's unstable, you cannot do multiple sorts on multiple keys  
[for example: list.sort(a.x > b.x).sort(a.y > b.y)]. To do this, use sort\_stable for all but the first sort  
[for example: list.sort(a.x > b.x).sort\_stable(a.y > b.y)] or combine the two sorts into one:  
[for example: list.sort(if (a.x != b.x), (a.x > b.x), (a.y > b.y)) ]

Returns: a sorted List of the same type as the given List.

Parameters: - a List of any type  
- an sorting expression which must return Logical

Example Usage:  
list(4, 6, 8, 2).sort(a < b); // returns list(2, 4, 6, 8)  
list("t", "h", "m").sort(a > b); // returns list("t", "m", "h")

Basic Functions	sort_stable ( List[Void], Expression )
-----------------	--

See the documentation for 'sort\_stable' for an example showing the difference between 'sort' and 'sort\_stable'.

### 3.2.5 sort\_stable ( List[Void], Expression )

'sort\_stable' is just like 'sort', except it uses a slow, but stable sorting algorithm that allows you to cascade sort's. That is, sort\_stable will ensure that the ordering of the equal elements in the input list is maintained in the output list.

sort\_stable orders a list based on an expression. sort processes a list until the sorting expression is true for all adjacent elements. Use 'a' and 'b' as element place holders in the expression.

Because 'sort\_stable' is so much slower than 'sort', use 'sort' whenever possible.

Returns: a sorted List of the same type as the given List.

Parameters:   - a List of any type  
                 - an sorting expression which must return Logical

Example Usage:

```
list(4, 6, 8, 2).sort_stable(a < b);   // returns list(2, 4, 6, 8)
list("t", "h", "m").sort_stable(a > b); // returns list("t", "m", "h")
```

@~ operations.sort(a.name < b.name).sort\_stable(a.category < b.category);  
Sorts operations using 'category' as the primary key, and 'name' as the secondary. The following does the same thing, but is over twice as fast:  
operations.sort(if (a.name != b.name, a.name < b.name, a.category < b.category));

@~Example showing the difference between sort and sort\_stable:

```
list(25, 26, 17, 18, 19).sort_stable(integer(a/10) < integer(b/10));
This sorts by the most significant digit, and returns list(17, 18, 19, 25, 26)
```

```
@~ list(25, 26, 17, 18, 19).sort(integer(a/10) < integer(b/10));
This sorts by the most significant digit, and returns list(19, 17, 18, 26, 25)
Notice that least-significant digits are in a different order.
```

Basic Functions	sublist ( List[Void], Integer, Integer )
-----------------	--

### 3.2.6 sublist ( List[Void], Integer, Integer )

sublist returns a contiguous portion of a list given a starting element and the desired size of the sublist. If the list does not contain as many elements as requested by the sublist, then the size of the resulting sublist is reduced.

Returns: a sublist of the given List; the sublist is the same type as the give List

Parameters:   - a List of anything  
                 - the index of an element in the given List;  
                 it becomes the first element of the resulting sublist;  
                 must be > 0  
                 - the number of elements in the resulting sublist;  
                 must be > 0

For example:

```
list(1, 2, 3, 4, 5).sublist(2, 3); // returns list(2, 3, 4)
list("a", "b", "c").sublist(3, 10); // returns list("c")
sublist(list(-1, 2, 0), 0, 0);   // returns list("")
list(1, 2, 3).sublist(0, 2);    // returns an error
list(1, 2, 3).sublist(1, 0);    // returns list("")
list(1, 2, 3).sublist(5, 2);    // returns list("")
```

### 3.2.7 unique ( List[Void] )

unique removes duplicate elements from a list.

WARNING: unique only works for data-types whose representation can be contained in a single word, so it does not work for types such as String, Period, Quantity, Time, and Unit. Since it is difficult for users to know which types will work, and which will not, try it and see. This limitation will be fixed someday.

Returns: a List of unique elements

Parameters:   - a List of single word data types

Example Usage:

```
unique(list(1, 2, 3, 2, 1);   // returns list(1, 2, 3)
```

3.2.8 contains ( List(Void), Void )

contains determines if a list contains a certain element. It returns true only if the 2nd parameter is an exact match for one of the elements in the list given as the first parameter.

Parameters:

- a List of any type
- the element to look for

Example Usage:

```
define(buffer1, sites.find("site1").buffers.find("one"));
define(buffer2, sites.find("site2").buffers.find("one"));
sites.find("site1").buffers.contains(buffer1); // Returns TRUE
sites.find("site1").buffers.contains(buffer2); // Returns FALSE
// It returns FALSE, because even though the key-fields for buffer1 and
// buffer2 are the same, they're different objects. An exact match must be found.
sites.find("site1").buffers.find(buffer2.name); // Returns TRUE

@~ list(1, 2, 3, 4).contains(0); // returns False
contains(list(3, 5, 7, 9), 7); // returns True
```

3.2.9 found ( List(Void), Void )

Given a list of models, the search uses the Model's key\_field, but only returns TRUE if the matching key-field model is found. It's just like find, only found only returns a logical instead of the element. For example, found(key) works just like find(key).exists, except it doesn't generate an error message when the key isn't found.

Returns: True when the given list contains an element with the same key-field; False otherwise.

Parameters:

- a List of any type
- the element or key\_field to look for (the key\_field must be the same as the Model's key\_field type)

Example Usage:

```
supply_chains.found("test"); // returns True if there is a supply
// chained named "test" in the supply
// chain list
```

3.2.10 element ( List(Void), Integer )

element returns the requested element from a list. Passing 1 in as the second argument returns the first element in the list.

Returns: The nth element of a List. The returned element retains its type. element reports an error if there is no such element.

Parameters:

- a List of any type
- requested element (an Integer > 0)

Example Usage:

```
element(list(5, 10, 15), 1); // returns 5
list("a", "b", "c").element(2); // returns "b"
list(1, 2, 3, 4).element(5); // reports an error
```

3.2.11 find ( List(Void), Void )

find searches a list of models and returns the first model whose key\_field matches a value. This version of find is used when the first argument is a list of models.

Returns: a Model with key\_field matching the given value. The model's type is retained.

find reports an error if no match is found.

Parameters:

- a Model List
- desired key\_field value (must be same type as the Model's key\_field type).

Example Usage:

```
supply_chains.find("test"); // returns the supply chain named "test"
sic.find("Dallas"); // returns the sic named "Dallas"
```

3.2.12 find\_or\_nonexistent ( List(Void), Void )

find searches a list of models and returns the first model whose key\_field matches a value. This version of find is used when the first argument is a list of models. This is exactly like find except nonexistent is returned when no match is found, instead of reporting an error.

Returns: a Model with key\_field matching the given value.

The model's type is retained.

Parameters:   - a Model List  
                - desired 'key\_field' value (must be same type as the Model's 'key\_field' type).

Example Usage:

```
supply_chains.find_or_nonexistent("test"): // returns nonexistent  
site.find_or_nonexistent("Dallas"): // returns the site named "Dallas"
```

3.2.13 find\_or\_create ( List[Void], Void )

find\_or\_create searches a list of models and either:  
\* returns the first model whose 'key\_field' matches a value, or  
\* if the model does not exist, creates and appends it to the end of the list followed by returning the newly created model.

find\_or\_create is only available to a list of models.

Returns: a Model with 'key\_field' matching the given value.  
The model's type is retained.

Parameters:   - a Model List,  
                - desired 'key\_field' value (must be same type as the Model's 'key\_field' type)

Example Usage:

In the following expressions, assume that the supply chain named "test" does not exist. Observe that the first expression yields an error because it did not find "test". The other expressions return the model named "test".

```
supply_chains.find("test");  
supply_chains.find_or_create("test");  
supply_chains.find("test");  
supply_chains.find_or_create("test");
```

3.2.14 list\_index ( List[Void], Void )

list\_index searches a list of models and returns the position in the list of the first model whose 'key\_field' matches a value. It also searches an arbitrary list for a match (using the '==' operator). list\_index reports an error if the list element is not found.

Returns: some\_list[some\_list.index(key)] is equivalent to some\_list.find(key)

Parameters:   - a List  
                - desired 'key\_field' value (must be same type as the Model's 'key\_field' type), or an element of the list.

Example Usage:

```
list("a", "b", "c").list_index("b") // returns 2
```

3.2.15 first ( List[Void] )

first returns the first element of a list.

Returns: The first element of the List. The returned element retains its type.

@-first returns NONEXISTENT if the List is empty.

Parameters:   - a List of any type

Example Usage:

```
first(list(5, 10, 15)): // returns 5  
list("one", "two", "three").first: // returns "one"
```

3.2.16 last ( List[Void] )

last returns the last element of a list.

Returns: The last element of the List. The returned element retains its type.

last returns NONEXISTENT if the List is empty.

Parameters:   - a List of any type

Example Usage:

```
last(list(5, 10, 15)): // returns 15  
list("one", "two", "three").last: // returns "three"
```

3.2.17 list ( )

list creates a new list of elements.

Returns: a List containing the passed in elements, in the order passed in.

Parameters: - variable; any number of elements of the same type  
Any type is valid. At least one element must be provided. Nonexistent parameters are ignored.

@~ An argument can itself be a list. However, list() will not create a list of lists. If an argument is a List, then its elements will be extracted and placed into the created list.

Example Usage:

```
list(3, list(2, 4, 6), 1); // returns a List of 5 Integers
// where "3" is the first element,
// "4" is the third element, and
// "1" is the fifth element.
list(symbol(a), symbol(b)); // returns a List of 2 symbols.
list(3, "three"); // returns an error since the given
// elements are different types
// (Integer and String).
list(); // returns an error (there must be at
list(make_type(none), string)); // returns an empty list(string)
```

3.2.18 integers ( Integer, Integer )

Generate a list of integers from min to max. This is used as a general looping mechanism when combined with for\_each.

Example:

```
integers(1, 5) -> integers(1, 5) = 1, 2, 3, 4, 5
integers(5, 4) -> integers(5, 4) =
integers(5, 5) -> integers(5, 5) = 5
integers(5, 6) -> integers(5, 6) = 5, 6
```

Returns: A list of integers

Parameters: - min, max

Example Usage: integers(1, 3).for\_each(echo(#));

3.3 Multi\_Keyed\_List Functions

Multi\_Keyed\_Lists are multi-hash-keyed collections of multiple elements of the same kind.

There are actually two types of multi-keyed-list:

- 1) Those created by multi\_key\_bucketize (see the multi\_key\_bucketize function for more detail)
- 2) Those created by multi\_keyed\_list (see the multi\_keyed\_list function for more detail)

An Multi\_Keyed\_Lists can be created, and queried via the functions in its group. Note: they may also be used by any of the generic List functions as well. (c.g. element, for\_each, num, etc.)

3.3.1

key ( Symbol, Symbol, Logical, Logical )  
@~Returns: a key definition for use by multi\_key

Parameters:

- name: the name of the key
- extract\_expr: an expression (usu. containing #) that returns a key value symbol
- filter\_expr: an expression (usu. containing #) that returns a logical indicating whether this key value will be excluded
- active: a logical that indicates whether this key is active or not

Example Usage:

```
define(key1, key("item", #.name, #.name != "leg", TRUE));
```

3.3.2

multi\_key ( Symbol, Void )  
@~Returns: a list of key definitions for use by multi\_keyed\_list and multi\_key\_bucketize

Parameters:

- type: the name of the type that this key definition will be used for, and one or more of the following
- key: a key definition (the output of the OIL function key)

Example Usage:

```
define(mk, multi_key("Item",
key1, // this is from the 'key' example
key("where", #.owner.name, #.owner.name != "CANADA", TRUE));
```

Basic Functions	multi_keyed_list ( List(Void), List(Void), Expression )
-----------------	---

3.3.3 multi\_keyed\_list ( List(Void), List(Void), Expression )

@~Returns: a multiply-keyed list of lists of elements

Parameters:

- values: a List of any type, for processing
- keys: a List of hash-key definitions (from multi\_key) and one or more of the following (but all of the same type)
- element: an expression that returns any type, evaluated for each element of values

Example Usage:

```
define(mkl, item_list, multi_keyed_list(mk, // this is from the 'multi_key' example
#name,
#category,
#delivery.name));
```

3.3.4 multi\_keyed\_element ( List(Void), Integer, Symbol, Symbol )

@~Returns: a list of the element(s) stored at the specified location(s)

Parameters:

- values: a List of any type, but created only by multi\_keyed\_list
- integer: the specific element in the list at that hash-keyed location to retrieve and one or more pairs of the following
- symbol: a key name.
- symbol: a key value.

Notes: - The number of specified key name-value pairs need not be as many as were used to build the original multi\_keyed\_list. Any keys not specified will be treated as if they were specified, but with a wildcard.

- Key name order does not need to be in the same order as it was when the list was built

Example Usage:

Here we get the list of all the #category that were stored for all site names == "USA"

```
mkl.multi_keyed_element(2, // mkl is from the multi_keyed_list example
```

Basic Functions	key_names ( List(Void) )
-----------------	--------------------------

"where", "USA");

3.3.5 key\_names ( List(Void) )

Use key\_names to get at the list of key names used to build this multi\_keyed\_list or multi\_keyed\_bucketized list.

Returns: a list of Symbols

Parameters:

- values: a Multi Keyed List of any type

Example Usage:

```
// mkl is from the multi_keyed_list example
mkl.key_names() = "item", "where"
```

3.3.6 key\_values ( List(Void), Symbol )

Use key\_values to get at the list of unique key values for the given name used to build this multi\_keyed\_list or multi\_keyed\_bucketized list.

Returns: a list of Symbols

Parameters:

- values: a Multi Keyed List of any type
- name: the name of a key used to create the list

Example Usage:

```
// mkl is from the multi_keyed_list example
mkl.key_values("item") = "table", "chair", "seat"
```

3.3.7 multi\_key\_bucketize ( List(Void), List(Void), List(Date\_Range), Expression, Expression )

Returns: a multiply-keyed and date\_ranged (bucketized) list of lists of elements

Parameters:

- values: a List of any type, for processing
- keys: a List of hash-key definitions (from multi\_key)
- date\_ranges: a List of contiguous date ranges (defines the buckets)
- date: an expression (usu. containing #) that returns a date, evaluated for each element of values
- and one or more of the following (but all of the same type)
- element: an expression (usu. containing #) that returns any type, evaluated for each element of values

Example Usage: (see bucketize and multi\_keyed\_list)

3.3.8 multi\_key\_bucketize\_element ( List(Void), Integer, Date, Symbol, Symbol )  
@~Returns: a list of the element(s) stored at the specified location(s)

Parameters:

- values: a List of any type, but created only by multi\_key\_bucketize
- integer: the specific element in the list at that hash-keyed location to retrieve
- date: an expression that returns a Date.
- and one or more pairs of the following
- symbol: a key name.
- symbol: a key value.

Notes: - The number of specified key name-value pairs need not be as many as were used to build the list in the original call to multi\_key\_bucketize. Any keys not specified will be treated as if they were specified, but with a wildcard.

- Key name order does not need to be in the same order as it was when the list was built

Example Usage: (see bucketize and multi\_keyed\_list)

3.4 Singly\_Keyed\_List Functions

@~Singly\_Keyed\_Lists are hash-keyed collections of elements of the same kind.

There are actually two types of singly-keyed-list:

- 1) Those created by bucketize (see the bucketize function for more detail)
- 2) Those created by keyed\_list (see the keyed\_list function for more detail)

An Singly\_Keyed\_List can be created and queried via the functions in its group.

Note: they may also be used by any of the generic List functions as well.  
(e.g. element, for\_each, num, etc.)

3.4.1 bucketize ( List(Void), List(Date\_Range), Expression, Expression )

Use bucketize when you want to be able to quickly retrieve values from a large list that is date-based in nature. It organizes the input list into a table with one column per date-range (from the 2nd parameter), and any number of row(s) indexed by a symbol). The intersection (table entries) are a list which you can access using the bucket\_list' function. The last two parameters are used to determine which of the input items go into which symbol/date\_range intersection list.

Returns: a list of values, organized for very quick retrieval characteristics.

Parameters:

- values: a List of any type
- date\_ranges: a List of Date\_Range
- date expression: an expression that returns a Date.
- symbol expression: an expression that returns a Symbol.

Description:

- 1) Takes the list of Date\_Range, and makes a bucket for each one (Note that in a list of Date\_Range buckets, that any two adjacent buckets, A and B, must have: A.end == B.start)
  - 2) Then it loops over the list of values and does the following for each:
    - 2.1) evaluates the date expression
    - 2.2) finds the bucket that the date expression fits in (i.e. bucket.start <= date < bucket.end) (Note this implies that given a bucket where bucket.start == bucket.end, nothing can ever fall within it)
    - 2.3) evaluates the symbol expression
    - 2.4) puts this value indexed by this symbol to the list in this bucket (NOTE: if more than one value is stored with the same key-symbol, then all values with that key-symbol will be returned)
  - by the single call to bucket\_list for that key-symbol.)
- Example Usage: take a list of Item\_Promises and put each one in the bucket whose date range contains the given date as evaluated in the expression #.owner.due.start, and hash-keys it with the given Symbol as evaluated in the expression #.item.name.



```
variable ips = delivery_promises_for_each(#.item_promises);  
variable bl = bucketize(date_range_list,  
    #.owner.date.start,  
    #.item.name);
```

Extended first time user example: Contrived and probably completely useless non-model example, but it shows in painstaking detail the flow of data:

```
define(date_range_list, list(date_range("95/01/01 00:00 / 95/01/02 00:00"), //  
    bucket#1  
    date_range("95/01/02 00:00 / 95/01/06 00:00"), // bucket#2  
    date_range("95/01/06 00:00 / 95/01/10 00:00"), // bucket#3  
    date_range("95/01/10 00:00 / 95/01/14 00:00"), // bucket#4  
    date_range("95/01/14 00:00 / 95/01/20 00:00"))) // bucket#5  
  
define(b_example, integers(5,10).bucketize(date_range_list,  
    date("95/01/00 00:00") + time(#.string & " day"),  
    #.string))
```

So for each of the integer numbers between 5 and 10, we generate a date for it, (Specifically: "95/01/05 00:00", "95/01/06 00:00", "95/01/07 00:00", "95/01/08 00:00",

"95/01/09 00:00", "95/01/10 00:00")

And therefore, the integer 5's value gets stored in bucket#2 since:

```
bucket#2.start <= (5's date) < bucket#2.end  
"95/01/02 00:00" <= "95/01/05 00:00" < "95/01/06 00:00"
```

Integer 6 ends up in bucket#3, (NOT in bucket#2, due to the non-inclusive nature of the end date of a bucket's defining

date-range)

Integers 7, 8 and 9 also get put in the list in bucket#3 as well.

Integer 10 ends up in bucket#4 (NOT in bucket#3, same reason as integer 6)

### 3.4.2 bucket\_list ( List(Void), Date, Symbol )

Use bucket\_list to quickly retrieve a specific value (or values) from a list created by bucketize.

Returns: a list of values

Parameters:

- values: a List of any type (but created by bucketize)
- date: a Date.
- Symbol: a Symbol.

Description:

1) Finds the bucket that the given Date falls within its Date\_Range (i.e. bucket.start <= date < bucket.end) 2) Retrieves any values that were stored in that bucket with a lookup key that matches the given Symbol.

Example Usage: (see bucketize for definition of b\_example)

```
b_example.bucket_list(date("95/01/10 00:00"), "9") --> nothing  
b_example.bucket_list(date("95/01/02 20:00"), "9") --> nothing
```

NOTE: These examples also show how the given lookup date does not have to match the

original date used to put the data in that specific bucket:

```
b_example.bucket_list(date("95/01/02 20:00"), "5") --> 5  
b_example.bucket_list(date("95/01/07 00:00"), "8") --> 8  
b_example.bucket_list(date("95/01/06 00:00"), "8") --> 8  
b_example.bucket_list(date("95/01/06 00:00"), "9") --> 9  
b_example.bucket_list(date("95/01/06 00:00"), "adam") --> nothing  
b_example.bucket_list(date("95/01/09 23:59"), "8") --> 8
```

### 3.4.3 bucket\_list\_by\_date ( List(Void), Date )

Use bucket\_list\_by\_date to quickly retrieve the whole list of values in the date\_range bucket that the given date falls within.

Returns: a list of values organized by key (i.e. the same kind of list as comes out of keyed\_list, and therefore usable by keyed\_element, and keys)

Parameters:

- values: a List of any type (but created by bucketize)
- date: a Date.

Description:

- 1) Finds the bucket that the given Date falls within its Date\_Range (i.e. bucket.start <= date < bucket.end)
- 2) Returns all the values that were stored in that bucket.

Example Usage: (see bucketize for definition of b\_example)

b\_example.bucket\_list\_by\_date(date("95/01/06 00:00")) --> 6, 7, 8, 9

#### 3.4.4 bucket\_list\_by\_key (List(Void), Symbol)

Use bucket\_list\_by\_key to quickly retrieve the whole list of values in all the date\_range buckets that match the given key.  
Returns: a list of values organized by key (i.e. the same kind of list as comes out of keyed\_list, and therefore usable by keyed\_element, and keys)

Parameters:

- values: a List of any type (but created by bucketize)
- symbol: a Symbol/String key.

Description:

- 1) Searches for the given key in every bucket in the list. 2) Returns the values found in a Keyed\_List that is keyed by the date\_range.string of the bucket it was found in.

@-Example Usage: (see bucketize for definition of b\_example)

b\_example.bucket\_list\_by\_key("8").keys() --> 95-01-12 00:00 / 95-01-20 00:00

Note: the default delimiter for date-ranges is a '.', not a '/', hence the above output. What the user should also derive from this is that given the two following expressions that seemingly should give the same output, only the first will give any output, due to the default date delimiter as mentioned above.

b\_example.bucket\_list\_by\_key("8").

keyed\_element(date\_range("95/01/06 00:00 / 95/01/10 00:00").string) --> 8

b\_example.bucket\_list\_by\_key("8").

keyed\_element("95/01/06 00:00 / 95/01/10 00:00") --> NOTHING

If you must do it that way, this will work:  
b\_example.bucket\_list\_by\_key("8").  
keyed\_element("95-01-06 00:00 / 95-01-10 00:00") --> 8  
However it is recommended to use date\_range().string to keep away from those types of subtleties.

#### 3.4.5 bucket\_symbols (List(Void))

Use bucket\_symbols to get at the list of unique Symbols (keys) that were used in building the given list during bucketize.

Returns: a list of Symbols

Parameters:

- values: a List of any type (but created by bucketize)

Example Usage: (see bucketize for definition of b\_example)  
b\_example.bucket\_symbols() --> "5", "6", "7", "8", "9", "10"

#### 3.4.6 keyed\_list (List(Void), Expression)

Use keyed\_list when you want to be able to quickly retrieve values from a large list

Returns: a list of values, organized for very quick retrieval characteristics.

Parameters:

- values: a List of any type
- symbol expression: an expression that returns a Symbol.

Description:

- 1) It loops over the list of values and does the following for each:
  - 1.1) evaluates the symbol expression
  - 1.2) stores this value, indexed by this symbol, in the new list. (NOTE: if more than one value is stored with the same key-symbol, then all values with that key-symbol will be returned by the single call to keyed\_element for that key-symbol.)

Example Usage:

Basic Functions	keyed_element (Symbol)
-----------------	------------------------

```
define(keyed, integers(1,10),keyed_list(#.string & "0"))
```

This one chooses to store the integer values:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10

with the respective key-symbols of:

"10", "20", "30", "40", "50", "60", "70", "80", "90", "100"

Use 'keyed\_element' to retrieve a value from a 'keyed\_list':

```
keyed.keyed_element("50"),sum -> 5
```

### 3.4.7 keyed\_element (Symbol)

Use 'keyed\_element' to quickly retrieve a specific value (or values) from a list created by 'keyed\_list'.

Returns: a list of values

Parameters:

- values: a List of any type (but created by 'keyed\_list')

- Symbol: a Symbol.

Description:

Retrieves any values that were stored in this list with a lookup key that matches the given Symbol. Retrieval is not done by a traditional linear-search, therefore as the lists get larger, the lookup speed of this method improves handsomely over linear-search.

Example Usage:

```
define(keyed, integers(1,10),keyed_list(#.string & "0"))
```

```
keyed.keyed_element("50") -> list(5)
```

```
keyed.keyed_element("5") -> nonexistent
```

### 3.4.8 keys (List(Void))

Use 'keys' to get at the list of unique Symbols (keys) that were used in building the given list during 'keyed\_list'.

Returns: a list of Symbols

Parameters:

Basic Functions	Recurse_List Functions
-----------------	------------------------

- values: a List of any type (but created by 'keyed\_list')

Example Usage:

```
define(keyed, integers(1,10),keyed_list(#.string & "0"))
```

```
keyed.keys() -> "10", "20", "30", "40", "50", "60", "70", "80", "90", "100"
```

## 3.5 Recurse\_List Functions

'Recurse\_Lists' are lists of elements created by 'recurse' or 'recurse\_or\_trim' and have the added feature of storing the recurse depth information. This can be handy for nesting/indentation.

Note: they may also be used by any of the generic List functions as well.

(e.g. element, for\_each, num, etc.)

### 3.5.1 recurse (List(Void), Expression)

'recurse' uses an expression to recursively processes each element in a list. The recursion ends when the expression returns an empty list.

'recurse' starts by placing each element in a list in a result list. Each element of the result list is then passed to the expression where is it substituted for the special placeholder '#'. This continues until the expression returns an empty list.

While executing 'expression', the variable 'parenis' is bound to the list of all the ancestors of '#'. The most recent ancestor is 'parenis.first', the root of the recursion is 'parenis.last'.

Returns: a List of the same type as the given list

Parameters: - a List of any type

- an expression which must return List's type

Example Usage:

Consider a List[Number] containing two Numbers { 12, 20 } and an Expression that returns the list of the factors of a Number, excluding itself. Calling 'recurse' on those will result in a List[Number]. The first Number in the resultant List will be "12". The Expression is evaluated with '#' set to "12" resulting in the List { 2 3 4 6 }. Then 'recurse' evaluates itself with the same Expression parameter and the newly generated List. So, the next Number in the output List is "2", { 12 2 ... }. Evaluating the Expression with '#' set to "2" results in an empty List, so the recursion stops. Thus, the next

element in the output list is "3", { 12 2 3 ...}. The Expression also returns empty List with "3", so the next element is "4", { 12 2 3 4 ...}. Evaluating the Expression with '#' set to "4" results in the List { 2 2 }. So, the next element in the result is "2", { 12 2 3 4 2 ...}. This continues until the final List { 12 2 3 4 2 2 6 2 3 20 2 4 2 2 5 10 2 5 } results.

recurse is particularly useful in the cell of a replicating Worksheet. In that context it has an additional effect: the depth of the recursion of each element is known. Various Layouts and Controls can take advantage of that information to let you see the "nesting", to let you see that the "2" and the "3" are under/within the "6", which, along with the "2", "3", and "4", is under/within the "12". For example, the indented\_general Control will indent proportional to the depth of the recurse that computed the value.

The best example of the use of recurse and indented\_general Control is the generation of an "indented bill-of-material". A function that gives the List of components of an Item gives a "single-level bill-of-material (BOM)". The recurse function allows the same function to be evaluated for each of those components to get the List of their components. And so on until the Items with no components are reached. Thus, recurse with the "single-level BOM" function generates a "multi-level BOM". By putting these in a column of a replicating Worksheet using a indented\_general Control, the Items will be indented according to their depth in the BOM. The result is an "indented BOM".

Limitations:

The result from 'recurse' is a very special list, with the indent information attached. If the list gets copied by 'filter' or 'for\_each' (or anything else) the indent info gets lost, which means the 'indented\_general' control will not indent. The work-around is to do all your filtering \*inside\* the recurse.

To avoid infinite recursions, 'recurse' is limited to a recursion depth of 1000, and it will not generate a list larger than 10,000 elements. You can change these limits with the 'recurse\_depth' and 'recurse\_items' options.

3.5.2 recurse\_and\_trim ( List|Void, Expression, Expression )  
recurse\_and\_trim is like the 'recurse' function, with an additional Logicalcan "filter" expression which is used to trim the resulting tree.

The 'filter' expression argument is a little different from the normal "filter" OIL function, because an element for which the expression returns "false" will still be included if one of its children's filter expressions returned "true". In other words, this filter is specifically designed to trim dead branches of the tree.

While executing both the recursion expression and the filter expression, the variable 'parents' is bound to the list of all the ancestors of '#'. The most recent ancestor is parents.first, the root of the recursion is parents.last.

Returns: a List of the same type as the given list  
Parameters: - a List of any type  
- an expression which must return List's type  
- an expression which returns a logical

3.5.3 current\_depth ( Cell )  
Returns the current depth of a (recursively) replicating cell  
Returns: An integer greater than or equal to zero  
Parameters: - A cell name or id

Example Usage:  
current\_depth(C1):  
3.6 Text\_String Functions  
Text strings are composed of zero or more characters. Text strings can be directly specified using pairs of double quotes. Nonprintable control characters can be specified by preceding them with a backslash (\). For instance, control-A is specified as "\a", control-B as "\b", etc. Here are some sample strings:

@ ~ "Dallas"  
"The quick brown fox jumped over the fence."  
"%\*#@#"  
"" (an empty string)  
"\"" (a string containing a single double quote)  
"\" (a string containing a single backslash)  
"Hello\nWorld!" (a string which embeds two control characters - a new line and a tab)

Strings can be created, edited, compared, and searched using the functions defined below.

3.6.1 string ( Void )  
string converts anything to a string, using the default format. The default format used depends on the argument's type.

Returns: a string

Parameters: - any type

Example Usage:

```
string(3.14);           // returns 3.14
string(quantily(100));  // returns 100
string(logical("yes")); // returns Yes
string("test");         // returns test
string(list(1, 2, 3));  // returns 1, 2, 3
```

### 3.6.2 string ( Void, Symbol )

string converts anything to a string, using the named format. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: a string

Parameters: - any type

- the named format (a Symbol) to use

Example Usage:

### 3.6.3 index ( String, String )

index returns the character index of the first occurrence of 'sub-string' in 'string'. If 'sub-string' is not found, the length of 'string' plus 1 is returned (the index of the character that is one past the end of the string). The character index is one based (i.e., the first character of a string is at index one).

Equivalent to index(string, sub-string, 1);

Returns: an integer

Parameters: - the String to search

- the sub-String to find

Example Usage:

```
index("one - two - three - " "on"); // returns "1"
index("--one - two -- three", "-"); // returns "6"
index("sub-string not found", "test"); // returns "21"
```

### 3.6.4 index ( String, String, Integer )

index returns the character index of the nth occurrence of 'sub-string' in 'string'. If 'sub-string' is not found, the length of 'string' plus 1 is returned (the index of the character that is one past the end of the string). The character index is one based (i.e., the first character of a string is at index one).

Returns: a String which contains either a character index or a string length

Parameters: - the String to search

- the sub-String to find

- the nth occurrence (an Integer); must be > 0

Example Usage:

```
index("one - ton - done - " "on", 3); // returns "14"
index("--one - two - three", "- ", 2); // returns "12"
index("sub-string not found", "test", 1); // returns "21"
```

### 3.6.5 left ( String )

left returns the first character of a string.

Equivalent to left(string, 1);

Returns: the first character of a string

Parameters: - the original String

Example Usage:

```
left("planer"); // returns "p"
```

### 3.6.6 left ( String, Integer )

left returns the first 'n' characters of a string. If 'n' is greater than the number of characters in the string, left returns the original string.

Returns: the first 'n' characters of a string

Parameters: - the original String

- the number of characters to return; must be >= 0

Example Usage:

Basic Functions	left (String, String )
-----------------	------------------------

```
left("planner", 4); // returns "plan"
left("abcde", 6); // returns "abcde"
left("test", 0); // returns ""
```

3.6.7 left (String, String )

left uses delimiters to split a string and returns the first, or leftmost, sub-string. If no delimiters are found, the original string is returned.

Equivalent to mid(string, delimiters, 1);

Returns: the leftmost sub-String or the original String

Parameters: - the original String  
- the delimiters (a String)

Example Usage:

```
left("one - two - three", "-"); // returns "one"
left("delimiter not found", "^"); // returns "delimiter not found"
```

3.6.8 lower (String )

lower converts all uppercase characters in a string to lowercase.

Returns: a String containing only lowercase characters

Parameters: - a String

Example Usage:

```
lower("Plan #35"); // returns "plan #35"
```

3.6.9 mid (String, Integer )

mid returns the nth character of a string. If the requested character is beyond the end of the string, mid returns the last character of the string.

Equivalent to mid(string, n, 1);

Returns: the requested character of a string or the last character of the string

Parameters: - the original String  
- the requested character (an Integer); must be > 0

Basic Functions	mid (String, Integer, Integer )
-----------------	---------------------------------

```
Example Usage:
mid("planning", 3); // returns "a"
mid("test", 6); // returns ""
```

3.6.10 mid (String, Integer, Integer )

mid returns the middle k characters starting at the nth character of a string. If the starting character is beyond the end of the string, mid returns an empty string. If the requested number of characters run beyond the end of the string, mid returns just the characters from the starting character to the end of the string.

Returns: the requested sub-String, or portion thereof, or an empty string

Parameters: - the original String  
- starting character (an Integer); must be > 0  
- the number of characters to return; must be >= 0

Example Usage:

```
mid("planner", 2, 3); // returns "lan"
mid("abcde", 4, 5); // returns "de"
mid("test", 6, 2); // returns ""
mid("test", 3, 0); // returns ""
```

3.6.11 mid (String, String, Integer )

mid uses delimiters to split a string and returns the requested sub-string. If no delimiters are found, mid returns an empty string.

Returns: the requested sub-String or an empty String

Parameters: - the original String  
- the delimiters (a String)  
- the requested sub-String (an Integer); must be > 0

Example Usage:

```
mid("one - two - three", "- ", 2); // returns "two"
mid("delimiter not found", "^", 3); // returns ""
```

3.6.12 proper (String )

proper capitalizes the first character of every word in a string, and converts the remain characters to lowercase. Specifically, proper capitalizes any letter not preceded by a letter. Otherwise, the letter is converted to lowercase. Underscores are replaced with spaces.

Basic Functions	justify (String, Integer, String)
-----------------	-----------------------------------

Returns: a proper String

Parameters: - the original String

Example Usage:  
proper("truly\_INTEGRATED planning"); // returns "Truly Integrated Planning"  
proper("this is a hard to read"); // returns "This is a Hard To Read".

### 3.6.13 justify (String, Integer, String)

The result is the first parameter with newlines inserted such that the width between newlines will be less than or equal to 'width' (the 2nd parameter). Lines may be broken at 'break\_on' characters (the 3rd parameter).

Returns: a string.

Parameters: - The original String.  
- The desired maximum width.  
- The set of characters on which to break the string.

Example Usage:  
some\_string.justify(40, ".,");

### 3.6.14 replace (String, Integer, Integer, String)

replace performs sub-string substitution. The replacement string can be any length (including the empty string). replace inserts text if the number of characters to replace is zero. replace deletes text if the replacement string is empty. replace appends text if the starting location is beyond the end of the string (number of characters is irrelevant). replace appends text to the beginning if the starting location is 1 and the number of characters is 0.

Returns: a String with the requested substitution

Parameters: - the original String  
- the starting location (an Integer); must be > 0  
- the number of characters to replace (an Integer); must be >= 0  
- the replacement String

Example Usage:  
replace("a master plan", 3, 6, "global"); // returns "a global plan"

Basic Functions	right (String, String)
-----------------	------------------------

```
replace("planning", 2, 4, ""); // returns "ping"
replace("planning", 6, 0, "erk"); // returns "planner king"
replace("plan", 6, 2, "ning"); // returns "planning"
replace("plan", 1, 0, "master"); // returns "master plan"
```

### 3.6.15 right (String, String)

right uses delimiters to split a string and returns the last, or rightmost, sub-string. If no delimiters are found, a copy of the original string is returned.

Equivalent to mid(string, delimiters, count(delimiters)).

Returns: the rightmost sub-String or a copy of the original String

Parameters: - the original String  
- the delimiters (a String)

Example Usage:  
right("one - two - three", "-"); // returns "three"  
right("delimiter not found", "a"); // returns "delimiter not found"

### 3.6.16 right (String)

right returns the last character of a string.

Equivalent to right(string, 1);

Returns: the last character of a string

Parameters: - the original String

Example Usage:

```
left("planner"); // returns "r"
```

### 3.6.17 right (String, Integer)

right returns the last 'n' characters of a string. If 'n' is greater than the number of characters in the string, right returns the original string.

Returns: the last 'n' characters of a string

Parameters: - the original String  
- the number of characters to return; must be >= 0

Example Usage:  
exp(0); // returns 1.0  
exp(-1); // returns 0.367879  
exp(2.71); // returns 15.0293

3.7.9 integer ( Number )  
integer converts a number to an integer, ignoring the number's fractional part.

Returns: the Integer portion of the given Number

Parameters: - a Number

Example Usage:  
integer(4.2); // returns 4  
integer(0.9); // returns 1  
integer(-4.6); // return -4

3.7.10 integer ( Percentage )  
integer converts a percentage to an integer. The percentage is truncated to the nearest integer.

Returns: the Integer representation of the Percentage

Parameters: - a Percentage

Example Usage:  
integer(percentage("25%")); // returns 0  
integer(percentage("285%")); // returns 2  
integer(percentage("99.5%")); // returns 99  
integer(percentage(0.25)); // returns 0

3.7.11 Integer ( String )  
integer converts a string to an integer using the default format.

Returns: the Integer representation of the String

Parameters: - a String; must only contain digits and/or a negative sign

Example Usage:  
number("22"); // returns 22

number("-100"); // returns -100  
number(""); // returns 0

3.7.12 integer ( String, Symbol )  
integer converts a string to an integer using the named format. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the Integer representation of the formatted String

Parameters: - a String  
- the named format (a Symbol) to use

Example Usage:  
3.7.13 ln ( Number )  
ln computes the natural (base e) logarithm of a number.

Returns: the natural logarithm of a Number

Parameters: - a Number; must be >= zero

Example Usage:  
ln(0.0); // returns -INFINITE  
ln(2.71); // returns 0.996949  
ln(-1.0); // returns an error

3.7.14 log ( Number )  
log computes the base-10 logarithm of a number. Note that log(x) is the same as log(x, 10) and log10(x).

Returns: the base-10 logarithm of a Number

Parameters: - a Number; must be >= zero

Example Usage:  
log(0.0); // returns -INFINITE  
log(10.0); // returns 1.0  
log(-1.0); // returns an error

3.7.15 log ( Number, Number )  
log computes the logarithm of a number to the specified base.



Basic Functions	log10 ( Number )
-----------------	------------------

Returns: the logarithm of a Number using the specified base

Parameters:   - a Number; must be >= zero  
              - a base (Number); must be > zero and != to 1

Example Usage:

```
log(0.0, 0.1); // returns -INFINITE
log(10.0, 5.0); // returns 1.43068
log(-1.0, 1.0); // returns an error
log(1.0, -1.0); // returns an error
```

3.7.16   log10 ( Number )

log10 computes the base-10 logarithm of a number. Note that log10(x) is the same as log(x, 10) and log(x).

Returns: the base-10 logarithm of a Number

Parameters:   - a Number; must be >= zero

Example Usage:

```
log(0.0);    // returns -INFINITE
log(10.0);   // returns 1.0
log(-1.0);   // returns an error
```

3.7.17   measure ( Quantity )

measure converts a quantity to a measure.

Returns: the Measure category of the given quantity.

Parameters:   - a Quantity

Example Usage:

```
measure(quantity("32"));    // returns unitless
measure(quantity("1 kg"));   // returns mass
measure(quantity("2 min"));   // returns time
```

3.7.18   measure ( String )

measure converts a string to a measure using the default format. The string must be a defined measure.

Basic Functions	measure ( String, Symbol )
-----------------	----------------------------

Returns: the Measure representation of the String or an error if not a defined measure

Parameters:   - a String; must be a defined measure

Example Usage:

```
measure("length");    // returns length
measure("time");       // returns time
measure("");           // returns unitless
measure("any");        // returns an error
measure("infinite");   // returns an error
```

3.7.19   measure ( String, Symbol )

measure converts a string to a measure using the named format. The string must be a defined measure. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the Measure representation of the String or an error if not a defined measure

Parameters:   - a String; must be a defined measure  
              - the named format (a Symbol) to use

Example Usage:

3.7.20   money ( String )

money converts a string to a money quantity using the default format. It is important to note that the money quantity can represent any specific national denomination, such as US dollars or Japanese yen.

Returns: the money Quantity representation of the String

Parameters:   - a String; the string must contain a numeric value

Example Usage:

```
money("0");    // returns 0.00
money("10.00"); // returns 10.00
money("6,66666"); // returns 6.67
money("$5");   // returns an error
```

3.7.21 money (String, Symbol)

money converts a string to a money quantity using the named format. It is important to note that the money quantity can represent any specific national denomination, such as US dollars or Japanese yen. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the money Quantity representation of the String in the named format

Parameters: - a String; the string must contain a numeric value  
- the named format (a Symbol) to use

Example Usage:

3.7.22 number (Integer)

number converts an integer to a number.

Returns: the Number representation of the given integer

Parameters: - an Integer

Example Usage:

```
number(0); // returns 0.0  
number(-1); // returns -1.0  
number(1000000); // returns 1000000.0
```

3.7.23 number (Percentage)

number converts a percentage to a number using the default format.

Returns: the Number representation of the given percentage.

Parameters: - a Percentage

Example Usage:

```
number(percentage("20%")); // returns 0.20  
number(percentage("33%")); // returns 0.33  
number(percentage("150%")); // returns 1.50
```

3.7.24 number (Quantity)

number converts a unitless quantity to a number.

Returns: the Number representation of the given unitless quantity.

Parameters: - a Quantity; must be unitless

Example Usage:

```
number(quantity("32")); // returns 32.0  
number(quantity("-1")); // returns -1.0
```

3.7.25 number (String)

number converts a string to a number using the default format. Strings "infinite" and "-infinite" are converted to positive and negative infinity respectively.

Returns: the Number representation of the String

Parameters: - a String; must only contain digits and/or a single decimal point and/or a negative sign

Example Usage:

```
number("22"); // returns 22.0  
number("3.1415926"); // returns 3.1415926  
number("0.000001"); // returns 0.000001  
number("infinite"); // returns INFINITE  
number(""); // returns 0.0
```

3.7.26 number (String, Symbol)

number converts a string to a number using the named format. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the Number representation of the formatted String

Parameters: - a String  
- the named format (a Symbol) to use

Example Usage:

3.7.27 percentage (Integer)  
percentage converts an integer to a percentage. Percentages are one-based (i.e., 1 = 100%).

Returns: the Percentage representation of the Integer

Parameters: - an Integer

Basic Functions	percentage ( Number )
-----------------	-----------------------

Example Usage:  
percentage(1); // returns 100%  
percentage(-10); // returns -1000%

3.7.28 percentage ( Number )  
percentage converts a number to a percentage. Percentages are one-based (i.e., 1 = 100%).

Returns: the Percentage representation of the Number

Parameters: - a Number

Example Usage:  
percentage(0.1); // returns 10%  
percentage(-1.01); // returns -101%

3.7.29 percentage ( Quantity )  
percentage converts a unitless quantity to a percentage. Percentages are one-based (i.e., 1 = 100%).

Returns: the Percentage representation of the Quantity or an error if the quantity is not unitless.

Parameters: - a unitless Quantity

Example Usage:  
percentage(quantity(2)); // returns 200%  
percentage(quantity(0.01)); // returns 1%  
percentage(quantity("1 m")); // returns an error

3.7.30 percentage ( String )  
percentage converts a string to a percentage using the default format.

Returns: the Percentage representation of the String

Parameters: - a String; the string must contain a numeric value

Example Usage:  
percentage("0"); // returns 0%  
percentage("10.0"); // returns 1000%

Basic Functions	percentage ( String, Symbol )
-----------------	-------------------------------

percentage("infinite"); // returns INFINITE  
percentage(""); // returns an error

3.7.31 percentage ( String, Symbol )  
percentage converts a string to a percentage using the named format. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the Percentage representation of the String

Parameters: - a String; the string must contain a numeric value  
- the named format (a Symbol) to use

Example Usage:

3.7.32 power ( Number, Number )  
power computes number^power (number raised to a power).

Returns: a number raised to a power

Parameters: - a Number to be raised  
- a power (Number) to raise to

Example Usage:  
power(10.0, 3.0); // returns 1000.0  
power(0.0, 0.0); // returns 1.0  
power(2.0, -2.5); // returns 0.176777

3.7.33 quantity ( String )  
quantity converts a computed string to a unitless quantity.

Returns: a unitless Quantity representation of the String

Parameters: - a String; the string must contain a numeric value

Example Usage:  
quantity("3.1415926"); // returns 3.1415926

3.7.34 quantity ( Integer )  
quantity converts an integer to a unitless quantity.

Basic Functions	quantity ( Number )
-----------------	---------------------

Returns: a unitless Quantity representation of the Integer

Parameters: - an Integer

Example Usage:

```
quantity(1); // returns 1
quantity(10); // returns 10
```

**3.7.35**    **quantity ( Number )**  
 quantity converts a number to a unitless quantity.

Returns: a unitless Quantity representation of the Number

Parameters: - a Number

Example Usage:

```
quantity(20); // returns 20
quantity(-2.2); // returns -2.2
```

**3.7.36**    **quantity ( String )**  
 quantity converts a string to a quantity using the default format. Quantities can have measurement units (e.g., feet, meters, centimeters, inches, kilograms, etc.), but this is not required. The measurement units must be a valid measure.

Returns: the Quantity representation of the String

Parameters: - a String; the string must contain a numeric value followed by an optional measurement unit

Example Usage:

```
quantity("22 m"); // returns 22 m
quantity("-5 kg"); // returns -5 kg
quantity("infinite"); // returns INFINITE
quantity("0 any"); // returns an error (any is not a valid
// measurement unit)
quantity(""); // returns an error
quantity("abc def"); // returns an error
```

Basic Functions	quantity ( String, Symbol )
-----------------	-----------------------------

**3.7.37**    **quantity ( String, Symbol )**  
 quantity converts a string to a quantity using the named format. Quantities can have measurements (e.g., feet, meters, centimeters, inches, kilograms, etc.), but this is not required. The measurements must be a valid. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the Quantity representation of the String in the named format

Parameters: - a String; the string must contain a numeric value followed by an optional measurement unit  
 - the named format (a Symbol) to use

Example Usage:

**3.7.38**    **quantity ( Time )**  
 quantity converts a time to a quantity.

Returns: a Quantity representation of the Time

Parameters: - a Time

Example Usage:  

```
quantity(time("3600 sec")); // returns 3600 sec
quantity(time("2 hr")); // returns 2 hr
```

**3.7.39**    **round ( Number )**  
 round returns the nearest integer to the given number.

Returns: the nearest Integer

Parameters: - a Number

Example Usage:  

```
round(9.9); // returns 10
round(0); // returns 0
round(-3.3); // returns -3
```

**3.7.40**    **round ( Number, Number )**  
 round returns the nearest multiple of a number to a given number.

Returns: the nearest multiple of a Number

Parameters: - a Number  
- a non-negative Number to multiply

Example Usage:  
round(9.9, 4.2); // returns 8.4  
round(0, 5); // returns 0  
round(5, 0); // returns an error  
round(-4.6, -1.0); // returns an error

3.7.41 round\_down ( Number )  
round\_down returns the largest integer smaller than a number. Rounds towards negative infinity.

Returns: -∞ < the largest Integer ≤ Number

Parameters: - a Number

Example Usage:  
round\_down(9.9); // returns 9  
round\_down(0.1); // returns 0  
round\_down(-0.1); // returns -1

3.7.42 round\_down ( Number, Number )  
round\_down returns the largest multiple of a number not greater than a given number. Rounds towards negative infinity.

Returns: -∞ < the largest multiple of a Number ≤ given Number

Parameters: - a Number

- a non-negative Number to multiply

Example Usage:  
round\_down(9.9, 4.2); // returns 8.4  
round\_down(0, 5); // returns 0  
round\_down(5, 0); // returns an error  
round\_down(-2.7, 1.5); // returns -3.0  
round\_down(-4.6, -1.0); // returns an error

3.7.43 round\_in ( Number )  
round\_in returns the next integer towards zero than a number.

Returns: the next Integer towards zero ≤ Number

Parameters: - a Number

Example Usage:  
round\_in(9.9); // returns 9  
round\_in(0.1); // returns 0  
round\_in(-0.1); // returns 0

3.7.44 round\_in ( Number, Number )  
round\_in returns the next multiple of a number towards zero than a given number.

Returns: the next multiple of a Number towards zero ≥ given Number

Parameters: - a Number

- a non-negative Number to multiply

Example Usage:  
round\_in(9.9, 4.2); // returns 8.4  
round\_in(0, 5); // returns 0  
round\_in(5, 0); // returns an error  
round\_in(-2.7, 1.5); // returns -1.5  
round\_in(-4.6, -1.0); // returns an error

3.7.45 round\_out ( Number )  
round\_out returns the next integer away from zero than a number.

Returns: the next Integer away from zero ≥ Number

Parameters: - a Number

Example Usage:  
round\_out(9.9); // returns 10  
round\_out(0.1); // returns 1  
round\_out(-0.1); // returns -1

3.7.46 round\_out ( Number, Number )  
round\_out returns the next multiple of a number away from zero than a given number.  
Returns: the next multiple of a Number away from zero ≥ given Number

Parameters:    - a Number  
                 - a non-negative Number to multiply

Example Usage:

```
round_out(9.9, 4.2); // returns 12.6
round_out(0, 5);    // returns 0
round_out(5, 0);    // returns an error
round_out(-2.7, 1.5); // returns -3.0
round_out(-4.6, -1.0); // returns an error
```

3.7.47 round\_up ( Number )

round\_up returns the smallest integer larger than a number. Rounds towards positive infinity.

Returns: +∞ > the smallest Integer >= Number

Parameters:    - a Number

Example Usage:

```
round_up(9.9); // returns 10
round_up(0.1); // returns 1
round_up(-0.1); // returns 0
```

3.7.48 round\_up ( Number, Number )

round\_up returns the smallest multiple of a number not less than a given number. Rounds towards positive infinity.

Returns: +∞ > the smallest multiple of a Number >= given Number

Parameters:    - a Number  
                 - a non-negative Number to multiply

Example Usage:

```
round_up(9.9, 4.2); // returns 12.6
round_up(0, 5);    // returns 0
round_up(5, 0);    // returns an error
round_up(-2.7, 1.5); // returns -1.5
round_up(-4.6, -1.0); // returns an error
```

3.7.49 sin ( Number )

sin computes the trigonometric sine of an angle, which is taken to be in radians.

Returns: the sine of an angle (a Number)

Parameters:    - an angle (a Number) specified in radians

Example Usage:

```
sin(0.0); // returns 0.0
sin(3.141592654); // returns 0.0
sin(-0.5); // returns -0.479426
```

3.7.50 sqrt ( Number )

sqrt computes the nonnegative square root of a number.

Returns: the nonnegative square root of a number

Parameters:    - a Number; must be >= zero

Example Usage:

```
sqrt(0.0); // returns 0.0
sqrt(4.0); // returns 2.0
sqrt(-2.0); // returns an error
```

3.7.51 tan ( Number )

tan computes the trigonometric tangent of an angle, which is taken to be in radians.

Returns: the tangent of an angle (a Number)

Parameters:    - an angle (a Number) specified in radians

Example Usage:

```
tan(0.0); // returns 0.0
tan(1.570796327); // returns -INFINITE
tan(-0.5); // returns -0.546302
```

3.7.52 time ( Quantity )

time converts a quantity to a time. The quantity must be expressed in valid time measurement units (sec, min, or hr).

Returns: the Time representation of the Quantity

Parameters: - a Quantity; the quantity must be expressed in a valid time measure

Example Usage:  
time(quantity("3 sec")); // returns 00:00:03  
time(quantity("-5 min")); // returns -00:05:00  
time(quantity("10")); // returns an error  
time(quantity("1 year")); // returns an error

3.7.53 time (String)  
time converts a string to a time using the default format. Times must specify measurement units. The valid units are sec, min, and hr.

Returns: the Time representation of the String

Parameters: - a String; the string must contain a numeric value followed by a valid time measure

Example Usage:  
time("22 sec"); // returns 00:00:23  
time("-5 min"); // returns -00:05:00  
time("3 hr"); // returns 03:00:00  
time("infinite"); // returns INFINITE  
time("22"); // returns an error  
time("0 any"); // returns an error (any is not a valid unit)  
time(""); // returns an error  
time("abc def"); // returns an error

3.7.54 time (String, Symbol)  
time converts a string to a time using the named format. Times must specify measurement units. The valid units are sec, min, and hr. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the Time representation of the String in the named format

Parameters: - a String; the string must contain a numeric value followed by a valid time measure  
- the named format (a Symbol) to use

Example Usage:

3.7.55 units (Quantity)  
units returns the quantity that represents one unit of a quantity.

Returns: quantity that represents one measure unit of a Quantity

Parameters: - a Quantity

Example Usage:  
units("0"); // returns "1"  
units("-2.5 kg"); // returns "1 kg"  
units("25 par/hr"); // returns "1 par/hr"

3.8 Random Functions  
Random Number generation functions.

3.8.1 rand\_integer (Integer, Integer)  
rand\_integer generates an evenly distributed random integer between two given integers, both inclusive. Use rand\_seed for initializing the pseudo-random-number generator used by this function.

Returns: a random Integer between two given values (both inclusive)

Parameters: - an inclusive Integer  
- an inclusive Integer

Example Usage:  
rand\_integer(0, 10); // returns a random integer >= 0 and <= 10  
rand\_integer(-1, 0); // returns either -1 or 0  
rand\_integer(1, 1); // returns 1  
rand\_integer(1, 0); // returns either 1 or 0

3.8.2 rand ( )  
rand generates an evenly distributed random number between 0 and 1, both inclusive. Use rand\_seed for initializing the pseudo-random-number generator used by this function.

Returns: a random Number between 0 and 1 (both inclusive).

Parameters: - N/A

Example Usage:

rand(): // returns a number between 0 and 1

3.8.3 rand ( Number, Number )

rand generates an evenly distributed random number between two given numbers, both inclusive. Use rand\_seed for initializing the pseudo-random-number generator used by this function.

Returns: a random Number between two given values (both inclusive)

Parameters: - an inclusive Number  
- an inclusive Number

Example Usage:

```
rand(0.5, 5.3); // returns a random integer >= 0.5 and <= 5.3  
rand(1.0, 1.0); // returns 1.0  
rand(2.0, -2.0); // returns a random number >= -2.0 and <= 2.0
```

3.8.4 rand\_seed ( Number )

rand\_seed initializes the pseudo-random-number generator using the given number as a seed. This allows for reproducibility of a series of random numbers generated from the rand functions.

Returns: N/A

Parameters: - a Number to be used as a seed

Example Usage:

```
rand_seed(39.0);
```

3.9 Quantity\_Range Functions

A Quantity\_Range is defined by a minimum quantity and a maximum quantity, both inclusive. The functions below can be used to create, query, and manipulate Quantity\_Ranges.

3.9.1 intersects ( Quantity\_Range, Quantity\_Range )

intersects determines if two quantity ranges overlap. The quantity ranges must have the same measurement units.

Returns: True if the two quantity ranges overlap; False otherwise.

Parameters: - a Quantity\_Range

- a Quantity\_Range

Example Usage:

Given...

```
a = quantity_range("1 ft / 3 ft");  
b = quantity_range("2 ft / 4 ft");  
c = quantity_range("4 ft / 6 ft");  
d = quantity_range("1 hr / 3 hr");
```

Then...

```
intersects(a, b); // returns True  
intersects(a, c); // returns False  
intersects(b, c); // returns True  
intersects(a, d); // returns an error
```

3.9.2 within ( Quantity, Quantity\_Range )

within determines if a quantity is within a quantity range. The quantity and quantity range must have the same measurement units.

Returns: True if the quantity is within the quantity range; False otherwise.

Parameters: - a Quantity  
- a Quantity\_Range

Example Usage:

Given...

```
a = quantity_range("1 ft / 3 ft");  
b = quantity_range("2 ft / 4 ft");  
c = quantity_range("4 ft / 6 ft");  
d = quantity_range("1 hr / 3 hr");
```

Then...

```
within("2 ft", b); // returns True  
within("2 ft", c); // returns True  
within("2 ft", c); // returns False  
within("2 ft", d); // returns an error
```



3.9.3 within ( Quantity\_Range, Quantity\_Range )

within determines if a quantity range is within another quantity range. The quantity ranges must have the same measurement units.

Returns: True if the first quantity range is within the second quantity range; False otherwise.

Parameters: - a Quantity\_Range  
- a Quantity\_Range

Example Usage:

Given...  
a = quantity\_range("1 ft / 3 ft");  
b = quantity\_range("5 ft / 6 ft");  
c = quantity\_range("4 ft / 8 ft");  
d = quantity\_range("1 hr / 3 hr");

Then...  
within(a, b); // returns False  
within(b, c); // returns True  
within(c, b); // returns False  
within(a, d); // returns an error

3.9.4 interval ( Quantity\_Range )

interval returns or sets the interval of a quantity range. When querying, it returns the interval as a quantity. When setting, it changes the maximum quantity of the quantity range.

Returns: a Quantity with the same measurement units as the Quantity\_Range

Parameters: - a Quantity\_Range

Example Usage:

Given...  
a = quantity\_range("1 hr / 3 hr");  
b = quantity\_range(0, "infinite");  
c = quantity\_range("30 sec / 180 sec");

Then...

3.9.5 limit ( Quantity, Quantity\_Range )

limit either returns a quantity if it is within a range or the closest boundary of the range. The quantity and the range must have the same measurement units.

Returns: a Quantity in the quantity range

Parameters: - a Quantity  
- a Quantity\_Range

Example Usage:

Given...  
a = quantity\_range("1 hr / 3 hr");

Then...  
limit("4 hr", a); // returns 4 hr  
limit("2 hr", a); // returns 2 hr  
limit("2 kg", a); // returns an error

3.9.6 limit ( Quantity\_Range, Quantity\_Range )

limit returns a quantity range which represents the intersection of two quantity ranges. If the two ranges do not intersect, then the maximum value of the first range is returned as a range. The two quantity ranges must have the same measurement units.

Returns: the intersection Quantity\_Range

Parameters: - a Quantity\_Range  
- a Quantity\_Range

Example Usage:

Given...  
a = quantity\_range("1 ft / 3 ft");  
b = quantity\_range("2 ft / 4 ft");  
c = quantity\_range("4 ft / 6 ft");  
d = quantity\_range("1 hr / 3 hr");

Then...  
limit(a, b); // returns 2 ft / 3 ft  
limit(a, c); // returns 3 ft / 3 ft  
limit(b, c); // returns 4 ft / 4 ft  
limit(a, d); // returns an error

3.9.7 max ( Quantity\_Range )  
max sets or returns the maximum quantity of a quantity range. If setting and the new maximum is less than the current minimum, then the range's minimum is changed to match the new maximum.

Returns: the maximum Quantity of a Quantity\_Range

Parameters: - a Quantity\_Range

Example Usage:

Given...  
a = quantity\_range("1 hr / 3 hr");  
b = quantity\_range(0, "infinite");  
c = quantity\_range("30 sec / 180 sec");

Then...  
max(a); // returns 3 hr  
max(b); // returns INFINITE  
c.max = "20 sec"; // changes the range to 20 sec / 20 sec

3.9.8 min ( Quantity\_Range )  
min sets or returns the minimum quantity of a quantity range. If setting and the new minimum is greater than the current maximum, then the range's maximum is changed to match the new minimum.

Returns: the minimum Quantity of a Quantity\_Range

Parameters: - a Quantity\_Range

Example Usage:

Given...  
a = quantity\_range("1 hr / 3 hr");

b = quantity\_range(0, "infinite");  
c = quantity\_range("30 sec / 180 sec");

Then...  
min(a); // returns 1 hr  
min(b); // returns 0  
c.min = "200 sec"; // changes the range to 200 sec / 200 sec

3.9.9 enclose ( Quantity, Quantity\_Range )  
enclose defines a new quantity range which includes a specified quantity. If the quantity is already within the given range, then the new range will match the given range. If quantity's measurement unit does not match the range's measurement unit, the original range is returned.

Returns: a Quantity\_Range with Quantity within it or the original Quantity\_Range if the Quantity has different measurement units

Parameters: - a Quantity to include  
- the Quantity\_Range to stretch

Example Usage:

Given...  
a = quantity\_range("1 hr / 3 hr");  
b = quantity\_range(0, "infinite");  
c = quantity\_range("30 sec / 180 sec");

Then...  
enclose("5 hr", a); // returns 1 hr / 5 hr  
enclose(100, b); // returns 0 / INFINITE  
enclose("10 sec", c); // returns 10 sec / 180 sec  
enclose("5 kg", a); // returns 1 hr / 3 hr

3.9.10 enclose ( Quantity\_Range, Quantity\_Range )  
enclose defines a new quantity range which includes the first range within the second range. If the first range is already within the second range, then the new range will match the second range. If the range's measurement units do not match, the second range is returned.

Returns: a Quantity\_Range with the first Quantity\_Range enclosed with the second Quantity range or the second Quantity\_Range if the first Quantity has different measurement units

Parameters: - a Quantity\_Range to include  
- the Quantity\_Range to stretch

Example Usage:

Given...

```
a = quantity_range("1 hr / 3 hr");
b = quantity_range("4 hr / 9 hr");
c = quantity_range("5 hr / 7 hr");
d = quantity_range("1 sec / 10 sec");
```

Then...

```
enclose(a, b); // returns 1 hr / 9 hr
enclose(b, c); // returns 4 hr / 9 hr
enclose(d, a); // returns 1 hr / 3 hr
```

3.9.11 quantity\_interval ( Quantity, Quantity )

quantity\_interval creates a quantity range using the first quantity as an anchor point and the second quantity as the interval from the anchor. If the second quantity is positive, the first quantity is the new range's minimum; if it is negative, the first quantity is the new range's maximum. The quantities must have the same measurement units.

Returns: a new Quantity\_Range

Parameters: - the anchor Quantity  
- the interval Quantity

Example Usage:

```
quantity_interval("1 kg", "3 kg"); // returns 1 kg / 4 kg
quantity_interval("1 kg", "0 kg"); // returns 1 kg / 1 kg
quantity_interval("4 kg", "-2 kg"); // returns 2 kg / 4 kg
quantity_interval("1 kg", "2 hr"); // returns an error
```

3.9.12 quantity\_range ( Quantity, Quantity )

quantity\_range creates a new quantity range given two quantities. The first quantity must be equal to or small than the second quantity. The quantities must have the same measurement unit.

Returns: a new Quantity\_Range

Parameters: - a Quantity; must be <= second Quantity  
- a Quantity

Example Usage:

```
quantity_range("1 hr", "4 hr"); // returns 1 hr / 4 hr
quantity_range("5 kg", "2 kg"); // returns an error
quantity_range("1 hr", "2 kg"); // returns an error
```

3.9.13 quantity\_range ( String )

quantity\_range converts a string into a quantity range using the default format. The string must contain two quantities which have the same measurement units and are separated by the string "<". The smaller quantity must be first.

Returns: the Quantity\_Range representation of the String

Parameters: - a String; the string must contain two quantities of the same measurement unit with the smaller one preceding the larger:

Example Usage:

```
quantity_range("1 hr / 4 hr"); // returns 1 hr / 4 hr
quantity_range("1 hr 4 hr"); // returns 1 hr / 1 hr
quantity_range("5 kg / 2 kg"); // returns an error
quantity_range("1 hr / 2 kg"); // returns an error
```

3.9.14 quantity\_range ( String, Symbol )

quantity\_range converts a string into a quantity range using the named format. The string must contain two quantities which have the same measurement units. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the Quantity\_Range representation of the String

Parameters: - a String; the string must contain two quantities of the same measurement unit.  
- the named format (a Symbol) to use

Example Usage:

3.9.15 set\_max ( Quantity\_Range, Quantity )

set\_max sets the maximum quantity of a quantity range. If the new maximum is less than the range's minimum, then the minimum is also set to the new maximum value. The quantity must have the same measurement units as the range.

Returns: the new maximum Quantity of Quantity\_Range

Parameters: - a Quantity\_Range  
- the maximum Quantity for the range

Example Usage:

Given...  
a = quantity\_range("1 hr / 3 hr");  
b = quantity\_range(0, "infinite");

Then...

set\_max(a, "2 hr"); // returns 1 hr / 2 hr  
set\_max(b, -2); // returns -2 / -2  
set\_max("5 kg", a); // returns an error

3.9.16 set\_min ( Quantity\_Range, Quantity )

set\_min sets the minimum quantity of a quantity range. If the new minimum is greater than the range's maximum, then the maximum is also set to the new minimum value. The quantity must have the same measurement units as the range.

Returns: the new minimum Quantity of Quantity\_Range

Parameters: - a Quantity\_Range  
- the minimum Quantity for the range

Example Usage:

Given...  
a = quantity\_range("1 hr / 3 hr");  
b = quantity\_range(0, "infinite");

Then...

set\_min(a, "5 hr"); // returns 5 hr / 5 hr  
set\_min(b, 100); // returns 100 / INFINITE

3.9.17 set\_interval ( Quantity\_Range, Quantity )

set\_interval adjusts the interval of a quantity range, increasing or decreasing the maximum quantity as necessary. The given quantity must be greater than zero and must have the same measurement units as the range.

Returns: an adjusted Quantity\_Range

Parameters: - a Quantity\_Range  
- a Quantity which defines a new range interval; must be > 0

Example Usage:

Given...  
a = quantity\_range("1 hr / 3 hr");  
b = quantity\_range(0, "infinite");

Then...

set\_interval(a, "5 hr"); // returns 1 hr / 6 hr  
set\_interval(b, 1); // returns 0 / 1  
set\_interval(a, "-1 hr"); // returns an error  
set\_interval(a, "3 kg"); // returns an error

3.10 Numeric\_List Functions

Numeric lists are sequential collections of numbers, integers, quantities, percentages, or times. The statistical functions documented within this section apply exclusively to numeric lists. They all return a single number, integer, quantity, percentage, or time based on the kind of numeric list they are given.

You can find the minimum or maximum in the List, or compute the sum, average, or standard deviation of the elements of the numeric list.

3.10.1 average ( List(Integer) )

average computes the average (sum / # elements) of all elements in an integer list. The list must have at least one element.

Returns: the average (a Number) of all the integers in a list

Parameters: - an integer List

Example Usage:



3.10.8 max ( List(Percentage) )

max returns the largest element of a percentage list. The list must have at least one element.

Returns: the largest Percentage in a list

Parameters: - a percentage List

Example Usage:

max(list(25%, 67%, 125%)); // returns 125%  
max(list()); // returns an error  
list(-50%, 10%, 65%).max; // returns 65%

3.10.9 max ( List(Quantity) )

max returns the largest element of a quantity list. The list must have at least one element.

Returns: the largest Quantity in a list

Parameters: - a quantity List

Example Usage:

max(list(25 kg, 67 kg, 125 kg)); // returns 125 kg  
max(list()); // returns an error  
list(-50 kg, 10 kg, 65 kg).max; // returns 65 kg

3.10.10 max ( List(Time) )

max returns the largest element of a time list. The list must have at least one element.

Returns: the largest Time in a list

Parameters: - a time List

Example Usage:

max(list(25 sec, 67 sec, 125 sec)); // returns 125 sec  
max(list()); // returns an error  
list(-50 sec, 10 sec, 65 sec).max; // returns 65 sec

3.10.11 min ( List(Integer) )

min returns the smallest element of an integer list. The list must have at least one element.

Returns: the smallest Integer in a list

Parameters: - an integer List

Example Usage:

min(list(1, 2, 3)); // returns 1  
min(list()); // returns an error  
list(-3, 8, -5).min; // returns -5

3.10.12 min ( List(Number) )

min returns the smallest element of a number list. The list must have at least one element.

Returns: the smallest Number in a list

Parameters: - a number List

Example Usage:

min(list(0.4, 1.6, 2.9)); // returns 0.4  
min(list()); // returns an error  
list(-3.3, 8.1, -5.7).min; // returns -5.7

3.10.13 min ( List(Percentage) )

min returns the smallest element of a percentage list. The list must have at least one element.

Returns: the smallest Percentage in a list

Parameters: - a percentage List

Example Usage:

min(list(25%, 67%, 125%)); // returns 25%  
min(list()); // returns an error  
list(-50%, 10%, 65%).min; // returns -50%

3.10.14 min ( List(Quantity) )

min returns the smallest element of a quantity list. The list must have at least one element.

Returns: the smallest Quantity in a list

Basic Functions	min ( List[Time] )
-----------------	--------------------

Parameters:    - a quantity List

Example Usage:

```
min(list(25 kg, 67 kg, 125 kg));    // returns 25 kg
min(list(-50 kg, 10 kg, 65 kg),min); // returns -50 kg
```

### 3.10.15    min ( List[Time] )

min returns the smallest element of a time list. The list must have at least one element.

Returns: the smallest Time in a list

Parameters:    - a time List

Example Usage:

```
min(list(25 sec, 67 sec, 125 sec));    // returns 25 sec
min(list());    // returns an error
list(-50 sec, 10 sec, 65 sec),min;    // returns -50 sec
```

### 3.10.16    product ( List[Integer] )

product multiplies together all the elements in an integer List. The list must have at least one element.

Returns: the product of all the integers in a list

Parameters:    - an integer List

Example Usage:

```
product(list(1, 2, 3)); // returns 6.0
product(list());    // returns an error
list(-3, 8, -5),product;    // returns 120.0
```

### 3.10.17    product ( List[Number] )

product multiplies together all the elements in a number List. The list must have at least one element.

Returns: the product of all the numbers in a list

Parameters:    - a number List

Basic Functions	product ( List[Percentage] )
-----------------	------------------------------

Example Usage:

```
product(list(0.4, 1.6, 2.9)); // returns 1.856
product(list());    // returns an error
list(-3.3, 8.1, -5.7),product; // returns 152.361
```

### 3.10.18    product ( List[Percentage] )

product multiplies together all the elements in a percentage List. The list must have at least one element.

Returns: the product of all the percentages in a list

Parameters:    - a percentage List

Example Usage:

```
product(list(25%, 67%, 125%)); // returns 20.9375%
product(list());    // returns an error
list(-50%, 10%, 65%),product; // returns -3.25%
```

### 3.10.19    product ( List[Quantity] )

product multiplies together all the elements in a quantity List. The list must have at least one element.

Returns: the product of all the quantities in a list

Parameters:    - a quantity List

Example Usage:

```
product(list(25 kg, 67 kg, 125 kg)); // returns 209375 kg
product(list());    // returns an error
list(-50 kg, 10 kg, 65 kg),product; // returns -32500 kg
```

### 3.10.20    stdev ( List[Integer] )

stdev computes the standard deviation of all elements in an integer list. The list must contain at least one element.

Returns: the standard deviation of all the integers in a list

Parameters:    - an integer List

Example Usage:

```
stdev(list(1, 2, 3)); // returns 1.0
```

```
stdev(l()); // returns an error  
list(-3, 8, -5).stdev; // returns 7.0
```

3.10.21 stdev (List(Number))

stdev computes the standard deviation of all elements in a number list. The list must contain at least one element.

Returns: the standard deviation of all the numbers in a list

Parameters: - a number List

Example Usage:  
stdev(list(0.4, 1.6, 2.9)); // returns 1.25033  
stdev(list()); // returns an error  
list(-3.3, 8.1, -5.7).stdev; // returns 7.37292

3.10.22 stdev (List(Percentage))

stdev computes the standard deviation of all elements in a percentage list. The list must contain at least one element.

Returns: the standard deviation of all the percentages in a list

Parameters: - a percentage List

Example Usage:  
stdev(list(25%, 67%, 125%)); // returns 0.502129  
stdev(list()); // returns an error  
list(-50%, 10%, 65%).stdev; // returns 0.575181

3.10.23 stdev (List(Quantity))

stdev computes the standard deviation of all elements in a quantity list. The list must contain at least one element.

Returns: the standard deviation of all the quantities in a list

Parameters: - a quantity List

Example Usage:  
stdev(list(25 kg, 67 kg, 125 kg)); // returns 50.21 kg  
stdev(list()); // returns an error  
list(-50 kg, 10 kg, 65 kg).stdev; // returns 57.52 kg

3.10.24 stdev (List(Time))

stdev computes the standard deviation of all elements in a time list. The list must contain at least one element.

Returns: the standard deviation of all the times in a list

Parameters: - a time List

Example Usage:  
stdev(list(25 sec, 67 sec, 125 sec)); // returns 50.21 sec  
stdev(list()); // returns an error  
list(-50 sec, 10 sec, 65 sec).stdev; // returns 57.52 sec

3.10.25 sum (List(Integer))

sum adds together all the elements in an integer list. The list must have at least one element.

Returns: the sum of all the integers in a list

Parameters: - an integer List

Example Usage:  
sum(list(1, 2, 3)); // returns 6  
sum(list()); // returns an error  
list(-3, 8, -5).sum; // returns 0

3.10.26 sum (List(Number))

sum adds together all the elements in a number list. The list must have at least one element.

Returns: the sum of all the numbers in a list

Parameters: - a number List

Example Usage:  
sum(list(0.4, 1.6, 2.9)); // returns 4.9  
sum(list()); // returns an error  
list(-3.3, 8.1, -5.7).sum; // returns -0.9



Basic Functions	sum ( List(Percentage) )
-----------------	--------------------------

3.10.27 sum ( List(Percentage) )  
 sum adds together all the elements in a percentage List. The list must have at least one element.

Returns: the sum of all the percentages in a list

Parameters: - a percentage List

Example Usage:  
 sum(list(25%, 67%, 125%)); // returns 217%  
 sum(list()); // returns an error  
 list(-50%, 10%, 65%).sum; // returns 25%

3.10.28 sum ( List(Quantity) )  
 sum adds together all the elements in a quantity List. The list must have at least one element.

Returns: the sum of all the quantities in a list

Parameters: - a quantity List

Example Usage:  
 sum(list(25 kg, 67 kg, 125 kg)); // returns 217 kg  
 sum(list()); // returns an error  
 list(-50 kg, 10 kg, 65 kg).sum; // returns 25 kg

3.10.29 sum ( List(Time) )  
 sum adds together all the elements in a time List. The list must have at least one element.

Returns: the sum of all the times in a list

Parameters: - a time List

Example Usage:  
 sum(list(25 sec, 67 sec, 125 sec)); // returns 217 sec  
 sum(list()); // returns an error  
 list(-50 sec, 10 sec, 65 sec).sum; // returns 25 sec

3.11 Date Functions  
 A date is a particular point in time, past, present, or future, with precision to one second. For example,

Basic Functions	max ( List(Date) )
-----------------	--------------------

1996 Mar 15 14:10:02 - March 15, 1996 at 2:10PM

Dates must be in the range from 1970 to 2050. Dates do not require a time, but including a time with a date further refines the exact moment the date represents. Dates can be subtracted, yielding the time duration between the two dates. Two special dates are supported:

infinite\_past("-----") - beginning of time  
 infinite\_future("+++++") - end of time

Dates can be created and queried with the following functions.

3.11.1 max ( List(Date) )  
 max returns the largest element of a date list. The list must have at least one element.

Returns: the largest Date in a list

Parameters: - a date List

Example Usage:  
 max(list(date("95/01/10 00:00"), date("95/01/05 00:00"))); // returns "95/01/10 00:00"

3.11.2 min ( List(Date) )  
 min returns the smallest element of a date list. The list must have at least one element.

Returns: the smallest Date in a list

Parameters: - a date List

Example Usage:  
 min(list(date("95/01/10 00:00"), date("95/01/05 00:00"))); // returns "95/01/05 00:00"

3.11.3 date ( String )  
 date converts a string to a date using the default format. The default format is "YY-MM-DD hh:mm:ss".  
 Returns: the Date representation of the String

Basic Functions	date (String, Symbol)
-----------------	-----------------------

Parameters: - a String; the string must contain a valid date and time, specified in seconds

Example Usage:

```
date("96-1-1 00:00:00"); // returns 96-01-01 00:00:00
date("97-2-29 12:00:00"); // returns an error
date("69-12-31 23:59:59"); // returns an error
date("51-1-1 00:00:00"); // returns an error
date("96-7-4"); // returns an error
date(""); // returns an error
date("abcdeF"); // returns an error
```

3.11.4 date (String, Symbol)  
date converts a string to a date using the named format. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the Date representation of the String

Parameters: - a String; the string must be able to be parsed using the named format  
- the named format (a Symbol) to use

Example Usage:

Given a 'minutes' format, specified as "MM-DD-YY hh:mm", then

```
@~ date("1-1-96 00:00", minutes); // returns 96-01-01 00:00:00
date("9-9-99", minutes); // returns an error
```

Given a 'no\_time' format, specified as "MM-DD-YY", then

```
@~ date("7-4-96", no_time); // returns 96-07-04 00:00:00
date("1-1-96 18:30:00", no_time); // returns an error
```

3.11.5 now ( )  
now returns the current date and time.

Returns: The current date and time

Parameters: - N/A

Example Usage:

Basic Functions	start_of_day (Date)
-----------------	---------------------

now() // returns the current date and time

3.11.6 start\_of\_day (Date)  
start\_of\_day returns the start of the day of a date in the local time zone.

Returns: The start of the day for the given date

Parameters: - a Date

Example Usage:

```
start_of_day("00-1-1 12:00:00"); // returns 00-01-01 00:00:00
```

3.11.7 start\_of\_day (Date, Integer)  
start\_of\_day returns the start of the day in the local time zone of a future date specified as a certain number of days beyond a given date.

Returns: The start of the day of a future date X number of days in the future

Parameters: - a Date  
- the number of days (Integer) in the future.  
Negative numbers indicate days in the past.

Example Usage:

```
start_of_day("00-1-28 14:30:23", 5); // returns 00-02-02 00:00:00
```

```
date_range(day_no, start_of_day(day_no, 1)) // a 1-day long date_range
Note: date_range(day_no, day_no + "24 hr") produces different results when day-
light-savings time changes.
```

3.11.8 start\_of\_month (Date)  
start\_of\_month returns the start of the month of a date in the local time zone.

Returns: The start of the month for the given date

Parameters: - a Date

Example Usage:

```
start_of_month("00-1-16 12:00:00"); // returns 00-01-01 00:00:00
```

Basic Functions	start_of_month (Date, Integer)
-----------------	--------------------------------

### 3.11.9 start\_of\_month (Date, Integer)

start\_of\_month returns the start of the month in the local time zone of a future or past date specified as a certain number of months beyond or prior to a given date.

Returns: The start of the month of a future or past date X number of months in the future or past

Parameters: - a Date  
- the number of months (Integer) in the future (if positive) or past (if negative)

Example Usage:

```
start_of_month("00-01-28 14:30:23", 5); // returns 00-06-01 00:00:00
start_of_month("97-02-01 00:00:00", -1); // returns 97-01-01 00:00:00
```

### 3.11.10 start\_of\_week (Date)

start\_of\_week returns the start of the week of a date in the local time zone. The start of a week is Monday.

Returns: The start of the week for the given date

Parameters: - a Date

Example Usage:

```
start_of_week("00-1-14 12:00:00"); // returns 00-01-10 00:00:00
```

### 3.11.11 start\_of\_week (Date, Integer)

start\_of\_week returns the start of the week in the local time zone of a future or past date specified as a certain number of weeks beyond or prior to a given date.

Returns: The start of the week of a future or past date X number of weeks in the future or past

Parameters: - a Date  
- the number of weeks (Integer) in the future (if positive) or past (if negative)

Example Usage:

```
start_of_week("00-1-28 14:30:23", 5); // returns 00-02-28 00:00:00
```

Basic Functions	start_of_year (Date)
-----------------	----------------------

```
start_of_week("00-1-28 14:30:23", -2); // returns 00-01-10 00:00:00
```

### 3.11.12 start\_of\_year (Date)

start\_of\_year returns the start of the year of a date in the local time zone.

Returns: The start of the year for the given date

Parameters: - a Date

Example Usage:

```
start_of_year("00-1-14 12:00:00"); // returns 00-01-01 00:00:00
```

### 3.11.13 start\_of\_year (Date, Integer)

start\_of\_year returns the start of the year in the local time zone of a future or past date specified as a certain number of years beyond or prior to a given date.

Returns: The start of the year of a future or past date X number of years in the future or past

Parameters: - a Date  
- the number of years (Integer) in the future (if positive) or past (if negative)

Example Usage:

```
start_of_year("00-1-28 14:30:23", 5); // returns 05-01-01 00:00:00
start_of_year("00-1-28 14:30:23", -2); // returns 98-01-01 00:00:00
```

### 3.11.14 horizon\_date (String)

horizon\_date converts a string to Horizon\_Date using the default format.

Returns: the Horizon\_Date representation of the String

Parameters: - a String

Example Usage:  
(Angle brackets indicate optional)  
(! symbol indicates either one or the other)  
(Time is optional for all the formats and can be specified in a similar way for all: hours:minutes, FORMAT)  
(FORMAT in the future at hours:minutes)

Basic Functions	horizon_date (String)
-----------------	-----------------------

(FORMAT at hour:minutes)  
where FORMAT can be specified as one of the following:

Days - N Days in the future

Num\_of\_days <days l day>  
horizon\_date("fifth day"); // returns 5th day

Absolute Day of nth Month

Num\_of\_days <days l day><ofl> Num\_of\_month monthmonths  
horizon\_date("2nd day of third month"); // returns 2nd of 3rd month

Relative day of nth month Num\_of\_days <daysday> and Num\_of\_month monthmonths  
horizon\_date("1 day and 4 months"); // returns 1st day and 4th month

Absolute day from end of Nth month

Num\_of\_days <daysday> <from> endlast <ofl>  
Num\_of\_month monthmonths  
horizon\_date("2 days from the end of the third month");

Nth day of week in future

<The> Num\_of\_occurrence <occurrence of the day> <Day\_of\_Week>  
horizon\_date("first Tuesday"); // returns 1st Tue

Day of week in Nth week of future

Day\_Of\_Week <ofl> Num\_Of\_Week WeekWeeksWk/wks  
<startstarts Day\_of\_Week\_start>  
horizon\_date("Tuesday of second week"); // returns Tue of 2nd week, start Mon

Nth Day\_of\_week of Nth month

Num\_of\_Week\_Day Day\_Of\_Week <ofl> Num\_of\_Month monthmonths  
horizon\_date("first Thursday of sixth month");

Day of week of nth week of nth month

Day\_Of\_Week <ofl> Num\_Of\_Week WeekWeeksWk/wks  
<ofl> Num\_Of\_Month monthmonths <startstarts Day\_Of\_Week>  
horizon\_date("Mo of 2nd week, third month"); // returns Mon, 2nd wk, 1st month, start Wed  
horizon\_date("Mo of first week, 1st month, week starts on Tuesday");

Use of time in horizon\_date:

Basic Functions	finite (Date)
-----------------	---------------

horizon\_date("10:30pm, 2nd day");  
horizon\_date("2nd day at 10:30pm");  
horizon\_date("2nd day in the future at 10:30pm");

### 3.11.15 finite (Date)

finite returns True if and only if the given date is finite

Returns: True when the Date is finite.

Parameters: - a Date

Example Usage:  
finite("00-6-14"); // returns True  
finite("++++"); // returns False

### 3.11.16 day\_of\_month (Date)

day\_of\_month returns the day of the month in the local time zone. The date must by finite.

Returns: the day of the month (1-31)

Parameters: - a Date

Example Usage:

day\_of\_month("00-1-14 12:00:00"); // returns 14

### 3.11.17 day\_of\_week (Date)

day\_of\_week returns the day of the week in the local time zone. The date must by finite.

Returns: the day of the week (1-7); 1 = Monday

Parameters: - a Date

Example Usage:

day\_of\_week("00-1-14 12:00:00"); // returns 5

### 3.11.18 day\_of\_year (Date)

day\_of\_year returns the day of the year in the local time zone. The date must by finite.

Basic Functions	hour ( Date )
-----------------	---------------

Returns: the day of the year (1-366)

Parameters: - a Date

Example Usage:

day\_of\_year("00-2-14 12:00:00"); // returns 45

### 3.11.19 hour ( Date )

hour returns the hour of the day in the local time zone. The date must by finite.

Returns: the hour of the day (0-23)

Parameters: - a Date

Example Usage:

hour("00-2-14 12:00:00"); // returns 12

### 3.11.20 minute ( Date )

minute returns the minute of the day in the local time zone. The date must by finite.

Returns: the minute of the day (0-59)

Parameters: - a Date

Example Usage:

minute("00-2-14 12:34:00"); // returns 34

### 3.11.21 month ( Date )

month returns the month of the date in the local time zone. The date must by finite.

Returns: the month of the date (1-12)

Parameters: - a Date

Example Usage:

month("00-2-14 12:34:00"); // returns 2

Basic Functions	number_of_weeks ( Date, Date )
-----------------	--------------------------------

### 3.11.22 number\_of\_weeks ( Date, Date )

number\_of\_weeks returns the number of weeks between two dates in the local time zone. If the first date occurs prior to the second date, the result is negative weeks.

Returns: the number of weeks

Parameters: - the reference Date  
- the actual Date

Example Usage:

number\_of\_weeks("00-1-1", "01/1/1"); // returns -52  
number\_of\_weeks("00-1-1", "99/7/4"); // returns 25

### 3.11.23 second ( Date )

second returns the second of the day in the local time zone. The date must by finite.

Returns: the second of the day (0-59)

Parameters: - a Date

Example Usage:

minute("00-2-14 12:34:13"); // returns 13

### 3.11.24 week\_of\_year ( Date )

week\_of\_year returns the number of weeks from the start of the date's year in the local time zone.

Returns: the number of weeks

Parameters: - a Date

Example Usage:

week\_of\_year("00-7-4"); // returns 27

### 3.11.25 year ( Date )

year returns the year of the date in the local time zone. The date must by finite.

Returns: the year of the date (1970-2050)

Basic Functions	within_daylight_savings_time ( Date )
-----------------	---------------------------------------

Parameters: - a Date

Example Usage:

month("00-2-14 12:34:00"); // returns 2000

3.11.26 within\_daylight\_savings\_time ( Date )  
within\_daylight\_saving\_time returns TRUE if and only if the given date falls within daylight saving time in the local time zone.

Returns: True when the Date is within daylight savings time;  
False otherwise.

Parameters: - a Date

Example Usage:  
If you are within the US Central time zone, then...

within\_daylight\_savings\_time("00-12-15"); // returns False  
within\_daylight\_savings\_time("00-6-14"); // returns True

3.11.27 within\_leap\_year ( Date )  
within\_leap\_year returns True if and only if the given date occurs during a leap year in the local time zone.

Returns: True when the Date is within a leap year; False otherwise

Parameters: - a Date

Example Usage:  
within\_leap\_year("00-6-14"); // returns True  
within\_leap\_year("98-6-14"); // returns False

3.11.28 enclosing\_day ( Date )  
enclosing\_day returns a day long time period, beginning at 00:00:00 on the given Date.

Returns: a day long time period (Date\_Range)

Parameters: - a Date

Basic Functions	date ( Horizon_Date, Date )
-----------------	-----------------------------

Example Usage:

enclosing\_day("00-6-14 00:00"); // returns 00-6-14 00:00 / 00-6-15 00:00  
enclosing\_day("00-6-15 18:10"); // returns 00-6-15 00:00 / 00-6-16 00:00

3.11.29 date ( Horizon\_Date, Date )  
Given a Horizon\_Date, and a start date, return the Date the given Date.

Returns: a day long time period (Date\_Range)

Parameters: - a Date

Example Usage: horizon\_date("Tuesday of second week"),date("95/02/01 02:00")->  
95-02-14 00:00

3.11.30 restriction ( String )  
restriction converts a string to Restriction using the default format. For more details, see the documentation for the Restriction format.

Returns: the Restriction representation of the String

Parameters: - a String

Example Usage: restriction("end before 98-03-25 12:30");

3.11.31 restriction ( String, Date )  
restriction converts a String and a Date to Restriction using the default format. restriction("end before", date) is equivalent to restriction("end before" & date.string). For more details, see the documentation for the Restriction format.

Returns: the Restriction representation of the String

Parameters: - a String and a Date

Example Usage: restriction("end before", now + "1 week");

3.11.32 restriction ( String, Date\_Range )  
restriction converts a String and a Date\_Range to Restriction using the default format. restriction("end in", period) is equivalent to restriction("end in" & period.string). For more details, see the documentation for the Restriction format.

Basic Functions	satisfies ( Restriction, Date_Range )
-----------------	---------------------------------------

Returns: the Restriction representation of the String

Parameters: - a String and a Date

Example Usage: restriction("in", date\_range(now + "1 day", now + "1 week"));

### 3.11.33 satisfies ( Restriction, Date\_Range )

Test to see if a Date\_Range satisfies a Restriction.

Returns: Returns TRUE if the Date\_Range satisfies the conditions specified by a Restriction.

Parameters: - a Restriction and a Date\_Range

Example Usage:

```
do(define(range, date_range(now + "1 day", now + "1 week")),
  satisfies(restriction("start first in", period), period)); // returns TRUE.
```

### 3.12 Date\_Range Functions

A Date\_Range defines a time period using an inclusive starting Date and an exclusive ending Date. The functions below can be used to create, query, and manipulate Date\_Ranges.

#### 3.12.1 date\_range ( Date, Date )

date\_range creates a new time period given two dates. The first date must occur before the second date.

Returns: a new time period (Date\_Range)

Parameters: - a Date

- a Date; must occur after first Date

Example Usage:

```
date_range("00-1-1 00:00", "00-2-1 00:00");
date_range("00-1-1 00:00", "00-1-1 00:00");
date_range("00-2-1 00:00", "00-1-1 00:00"); // returns an error
```

#### 3.12.2 date\_range ( Date, Time )

date\_range creates a new time period given a date and a time interval. If the time interval is positive, the date is the starting date. If the time interval is negative, the date is the ending date.

Returns: a new time period (Date\_Range)

Basic Functions	date_range ( String )
-----------------	-----------------------

Parameters: - a Date

- a Time interval

Example Usage:

```
date_range("00-1-1 00:00", "1 week"); // returns 00-01-01 00:00 <
date_range("00-2-14 00:00", "-1 week"); // returns 00-02-07 00:00 <
// 00-02-14 00:00
```

#### 3.12.3 date\_range ( String )

date\_range converts a string into a time period using the default format. The string must contain two dates with the early date preceding the later date.

Returns: the Date\_Range representation of the String

Parameters: - a String; the string must contain two dates with the early one first in the following format: "YY-MM-DD hh:mm YY-MM-DD hh:mm"

Example Usage:

```
date_range("00-1-1 00:00 / 00-1-5 00:00");
date_range("97-9-4 00:00 97-10-1 00:00");
date_range("98-6-1 00:00 / 98-5-3 00:00"); // returns an error
date_range("00-1-1 00:00 / 00:00"); // returns an error
```

#### 3.12.4 date\_range ( String, Symbol )

date\_range converts a string into a time period using the named format. The string must contain two dates. The named format must be appropriate for the string. Otherwise, the conversion will not work.

Returns: the Date\_Range representation of the String

Parameters: - a String; the string must contain two dates

- the named format (a Symbol) to use

Example Usage:

#### 3.12.5 enclose ( Date, Date\_Range )

enclose adjusts a time period to include a date. If the time period already covers the date, then no adjust is made.

Returns: an encompassing time period (Date\_Range)

Basic Functions	enclose ( Date_Range, Date_Range )
-----------------	------------------------------------

Parameters: - the Date to include in a time period  
- the time period (Date\_Range) to adjust

Example Usage:

Given...  
a = date\_range("00-1-1 00:00", "00-3-9 00:00");

Then...

```
enclose("00-4-1 00:00", a); // returns 00-01-01 00:00 / 00-04-01 00:00
enclose("00-2-3 00:00", a); // returns 00-01-01 00:00 / 00-03-09 00:00
enclose("99-11-3 00:00", a); // returns 99-11-03 00:00 / 00-03-09 00:00
```

### 3.12.6 enclose ( Date\_Range, Date\_Range )

enclose adjusts a time period to include another time period. If the time period already covers the other time period, then no adjustment is made.

Returns: an encompassing time period (Date\_Range)

Parameters: - the time period (Date\_Range) to cover  
- the time period (Date\_Range) to adjust

Example Usage:

Given...  
a = date\_range("00-1-1 00:00", "00-3-9 00:00");  
b = date\_range("99-12-15 00:00", "99-12-25 00:00");  
c = date\_range("00-5-1 00:00", "00-8-30 00:00");  
d = date\_range("99-11-1 00:00", "00-1-31 00:00");

Then...

```
enclose(b, a); // returns 99-12-15 00:00 / 00-03-09 00:00
enclose(c, a); // returns 00-01-01 00:00 / 00-08-30 00:00
enclose(b, d); // returns 99-11-01 00:00 / 00-01-31 00:00
enclose(d, a); // returns 99-11-01 00:00 / 00-03-09 00:00
```

### 3.12.7 end ( Date\_Range )

end either sets or returns the ending date of a time period. When setting, if the end date occurs before the starting date, then the start date is adjusted to match the new ending date.

Basic Functions	intersects ( Date_Range, Date_Range )
-----------------	---------------------------------------

Returns: the ending Date of a time period

Parameters: - a time period (Date\_Range)

Example Usage:

Given...  
a = date\_range("00-1-1 00:00:00 / 00-4-15 00:00:00");

Then...

```
end(a); // returns 00-04-15 00:00:00
a.end = "00-2-10 00:00:00"; // changes the range to
// 00-01-01 00:00:00 / 00-02-10 00:00:00
a.end = "99-6-2 00:00:00"; // changes the range to
// 99-06-02 00:00:00 / 99-06-02 00:00:00
```

### 3.12.8 intersects ( Date\_Range, Date\_Range )

intersects determines if two time periods overlap.

Returns: True if the two time periods overlap; False otherwise.

Parameters: - a time period (Date\_Range)  
- a time period (Date\_Range)

Example Usage:

Given...  
a = date\_range("00-1-1 00:00", "00-3-9 00:00");  
b = date\_range("99-12-15 00:00", "99-12-25 00:00");  
c = date\_range("00-5-1 00:00", "00-8-30 00:00");  
d = date\_range("99-11-1 00:00", "00-1-31 00:00");

Then...

```
intersects(b, a); // returns False
intersects(c, a); // returns False
intersects(b, d); // returns True
intersects(d, a); // returns True
```



Basic Functions	limit ( Date, Date_Range )
-----------------	----------------------------

**3.12.9 limit ( Date, Date\_Range )**  
limit either returns a date if it is within a time period or the closest boundary of the time period.

Returns: a Date within the time period

Parameters: - a Date  
- a time period (Date\_Range)

Example Usage:

Given...  
a = date\_range("97-3-14 00:00:00 / 97-5-20 00:00:00");

Then...  
limit("97-4-26 00:00:00", a); // returns 97-04-26 00:00:00  
limit("97-2-1 00:00:00", a); // returns 97-03-14 00:00:00  
limit("97-8-4 00:00:00", a); // returns 97-05-20 00:00:00

**3.12.10 limit ( Date\_Range, Date\_Range )**

limit returns a time period which represents the intersect of two time periods. If the two time periods do not intersect, then the first time period's maximum value is returned as a time period.

Returns: the intersection time period (Date\_Range)

Parameters: - a time period (Date\_Range)  
- a time period (Date\_Range)

Example Usage:

Given...  
a = date\_range("00-1-1 00:00", "00-3-9 00:00");  
b = date\_range("99-12-15 00:00", "99-12-25 00:00");  
c = date\_range("00-5-1 00:00", "00-8-30 00:00");  
d = date\_range("99-11-1 00:00", "00-1-31 00:00");

Then...  
limit(b, a); // returns 99-12-25 00:00 / 99-12-25 00:00  
limit(c, a); // returns 00-08-30 00:00 / 00-08-30 00:00  
limit(b, d); // returns 99-11-01 00:00 / 99-12-25 00:00

Basic Functions	set_end ( Date_Range, Date )
-----------------	------------------------------

limit(d, a); // returns 00-01-01 00:00 / 00-01-31 00:00

**3.12.11 set\_end ( Date\_Range, Date )**  
set\_end defines the end date of a time period. If the new end date occurs prior to the time period's start date, then the time period's start date is also set the new end date.

Returns: a time period's end Date

Parameters: - a time period (Date\_Range)  
- a new end Date

Example Usage:

Given...  
a = date\_range("99-12-15 00:00", "99-12-25 00:00");  
b = date\_range("00-5-1 00:00", "00-8-30 00:00");

Then...  
set\_end(a, "99-12-20 00:00"); // returns 99-12-20 00:00  
set\_end(b, "00-9-30 00:00"); // returns 00-09-30 00:00  
set\_end(a, "99-11-1 00:00"); // returns 99-11-01 00:00

**3.12.12 set\_start ( Date\_Range, Date )**

set\_start defines the start date of a time period. If the new start date occurs after to the time period's end date, then the time period's end date is also set the new start date.

Returns: a time period's start Date

Parameters: - a time period (Date\_Range)  
- a new start Date

Example Usage:

Given...  
a = date\_range("99-12-15 00:00", "99/12/25 00:00");  
b = date\_range("00-5-1 00:00", "00/8/30 00:00");

Then...  
set\_start(a, "99-12-20 00:00"); // returns 99-12-20 00:00  
set\_start(b, "00-4-1 00:00"); // returns 00-04-01 00:00  
set\_start(a, "99-12-28 00:00"); // returns 99-12-28 00:00

3.12.13 set\_time ( Date\_Range, Time )

set\_time defines a time period's duration, moving the end date as necessary. The duration must be greater than or equal to zero.

Returns: the time period's duration (a Time)

Parameters: - a time period (Date\_Range)  
- the new duration (Time); must be >= 0

Example Usage:

Given...

```
a = date_range("99-12-15 00:00", "99-12-25 00:00");  
b = date_range("00-5-1 00:00", "00-8-30 00:00");
```

Then...

```
set_time(a, "7 day"); // returns 7 days; a's end date is changed  
// to 199-2-22 00:00  
set_time(b, "8 hr"); // returns 8 hr; b's end date is changed  
// to 00-05-01 08:00  
set_time(a, "-2 day"); // returns an error
```

3.12.14 start ( Date\_Range )

start either defines or returns a time period's start date. When defining, if the new start date occurs after to the time period's end date, then the time period's end date is also set the new start date.

Returns: a time period's start Date

Parameters: - a time period (Date\_Range)

Example Usage:

Given...

```
a = date_range("99-12-15 00:00", "99-12-25 00:00");  
b = date_range("00-5-1 00:00", "00-8-30 00:00");
```

Then...

```
start(a, "99-12-20 00:00"); // returns 99-12-20 00:00  
start(b, "00-4-1 00:00"); // returns 00-04-01 00:00
```

```
a.start = "99-12-28 00:00"; // returns 99-12-28 00:00
```

3.12.15 time ( Date\_Range )

time either defines or returns a time period's duration. When defining, the time period's end date is moved as necessary. Also, the duration must be greater than or equal to zero.

Returns: the time period's duration (a Time)

Parameters: - a time period (Date\_Range)

Example Usage:

Given...

```
a = date_range("99-12-15 00:00", "99-12-25 00:00");  
b = date_range("00-5-1 00:00", "00-8-30 00:00");
```

Then...

```
time(a, "7 day"); // returns 7 day  
time(b, "8 hr"); // returns 8 hr  
a.time = "5 day"; // returns 5 day; a's end date is changed to  
// 99-12-20 00:00  
b.time = "-8 hr"; // returns an error
```

3.12.16 within ( Date, Date\_Range )

within determines if a date falls within a time period.

Returns: True if the Date is within the time period; False otherwise.

Parameters: - a Date  
- a time period (Date\_Range)

Example Usage:

Given...

```
a = date_range("00-1-1 00:00", "00-3-9 00:00");
```

Then...

```
within("00-4-1 00:00", a); // returns False  
within("00-2-3 00:00", a); // returns True  
within("99-11-3 00:00", a); // returns False
```

3.12.17 within ( Date\_Range, Date\_Range )

within determines if a time period falls completely within another time period.

Returns: True if the first time period falls completely within the second time period; False otherwise

Parameters: - a time period (Date\_Range)  
- a time period (Date\_Range)

Example Usage:

Given...  
a = date\_range("00-1-1 00:00", "00-3-9 00:00");  
b = date\_range("99-12-15 00:00", "99-12-25 00:00");  
c = date\_range("00-5-1 00:00", "00-8-30 00:00");  
d = date\_range("99-11-1 00:00", "00-1-31 00:00");

Then...

within(b, a); // returns False  
within(c, a); // returns False  
within(b, d); // returns True  
within(d, a); // returns False

3.13 Date\_Range\_List Functions

Date\_Range\_List are lists of consecutive time periods (date ranges). The below functions generate consecutive time periods (Date\_Ranges), with no gaps, suitable for use as horizons for planning.

3.13.1 days ( Date\_Range )

days creates a list of consecutive one day time periods which span the given time period. Each time period starts and ends on day boundaries (00:00), expect perhaps the start of the first and end of the last day, which will be the start and end time of the given time period.

Returns: A List of consecutive one day Date\_Ranges with no gaps, spanning the requested time period.

Parameters: - the time period to span (Date\_Range)

Example Usage:

Given...

a = date\_range("97-06-01 06:00", "97-06-03 14:00");  
b = date\_range("97-06-10 02:00", "97-06-10 22:00");  
c = date\_range("97-06-15 00:00", "97-06-17 00:00");

Then...

days(a); // returns a list containing three time periods:  
// { 97-06-01 06:00 / 97-06-02 00:00,  
// 97-06-02 00:00 / 97-06-03 00:00,  
// 97-06-03 00:00 / 97-06-03 14:00 }

days(b); // returns a list containing one time period:  
// { 97-06-10 02:00 / 97-06-10 22:00 }

days(c); // returns a list containing two time periods:  
// { 97-06-15 00:00 / 97-06-16 00:00,  
// 97-06-16 00:00 / 97-06-17 00:00 }

3.13.2 days ( Date\_Range, Time )

days creates a list of consecutive one day time periods which span the given time period. Each time period starts and ends on day boundaries plus the given time offset except perhaps the start of the first and end of the last day, which will be the start and end time of the given time period.

Returns: A List of consecutive one day Date\_Ranges with no gaps, spanning the requested time period.

Parameters: - the time period to span (Date\_Range)  
- the starting Time of each time period; must be >= 0  
if greater than 24 hours, then first time period will span more than a single day.

Example Usage:

Given...  
a = date\_range("97-06-01 06:00", "97-06-03 14:00");  
b = date\_range("97-06-10 02:00", "97-06-11 22:00");  
c = date\_range("97-06-15 00:00", "97-06-20 00:00");

Then...

days(a, "4 hr"); // returns a list containing three time periods:

```
// { 97-06-01 06:00 / 97-06-02 04:00,
// 97-06-02 04:00 / 97-06-03 04:00,
// 97-06-03 04:00 / 97-06-03 14:00 }

days(b, "600 s"); // returns a list containing two time periods:
// { 97-06-10 02:00 / 97-06-11 00:10,
// 97-06-11 00:10 / 97-06-11 22:00 }

days(c, "48 hr"); // returns a list containing three time periods:
// { 97-06-15 00:00 / 97-06-18 00:00,
// 97-06-18 00:00 / 97-06-19 00:00,
// 97-06-19 00:00 / 97-06-20 00:00 }
```

days(a, "- 1 hr"); // returns an error

3.13.3 months ( Date\_Range )

months creates a list of consecutive one month time periods which span the given time period. Each time period starts and ends on month boundaries (first day of the month at 00:00), expect perhaps the start of the first and end of the last month, which will be the start and end time of the given time period.

Returns: A List of consecutive one month Date\_Ranges with no gaps, spanning the requested time period.

Parameters: - the time period to span (Date\_Range)

Example Usage:

Given...

```
a = date_range("97-06-01 06:00", "97-08-03 14:00");
b = date_range("97-06-10 02:00", "97-06-29 22:00");
c = date_range("97-06-15 00:00", "97-07-17 00:00");
```

Then...

```
months(a); // returns a list containing three time periods:
// { 97-06-01 06:00 / 97-07-01 00:00,
// 97-07-01 00:00 / 97-08-01 00:00,
// 97-08-01 00:00 / 97-08-03 14:00 }

months(b); // returns a list containing one time period:
// { 97-06-10 02:00 / 97-06-29 22:00 }
```

```
months(c); // returns a list containing two time periods:
// { 97-06-15 00:00 / 97-07-01 00:00,
// 97-07-01 00:00 / 97-07-17 00:00 }
```

3.13.4 quarters ( Date\_Range )

quarters creates a list of consecutive one quarter time periods which span the given time period. Each time period starts and ends on quarter boundaries (first day of the first, fourth, seventh, or ninth month at 00:00), expect perhaps the start of the first and end of the last quarter, which will be the start and end time of the given time period.

Returns: A List of consecutive one quarter Date\_Ranges with no gaps, spanning the requested time period.

Parameters: - the time period to span (Date\_Range)

Example Usage:

Given...

```
a = date_range("97-06-01 06:00", "97-10-03 14:00");
b = date_range("97-07-10 02:00", "97-09-01 22:00");
c = date_range("97-06-15 00:00", "97-07-17 00:00");
```

Then...

```
quarters(a); // returns a list containing three time periods:
// { 97-06-01 06:00 / 97-07-01 00:00,
// 97-07-01 00:00 / 97-10-01 00:00,
// 97-10-01 00:00 / 97-10-03 14:00 }

quarters(b); // returns a list containing one time period:
// { 97-07-10 02:00 / 97-09-01 22:00 }

quarters(c); // returns a list containing two time periods:
// { 97-06-15 00:00 / 97-07-01 00:00,
// 97-07-01 00:00 / 97-07-17 00:00 }
```

3.13.5 shifts ( Date\_Range )

shifts creates a list of consecutive one shift time periods which span the given time period. Each time period starts and ends on shift boundaries, expect perhaps the start of the first and end of the last shift, which will be the start and end time of the given time period. Shifts default to 8 hours starting at 00:00.

Returns: A List of consecutive one shift Date\_Ranges with no gaps, spanning the requested time period.

Parameters: - the time period to span (Date\_Range)

Example Usage:

Given...

```
a = date_range("97-06-01 06:00", "97-06-01 23:00");
b = date_range("97-06-10 09:00", "97-06-10 15:00");
c = date_range("97-06-15 00:00", "97-06-15 12:00");
```

Then...

shifts(a): // returns a list containing three time periods:

```
// { 97-06-01 06:00 / 97-06-01 08:00,
// 97-06-01 08:00 / 97-06-01 16:00,
// 97-06-01 16:00 / 97-06-01 23:00 }
```

shifts(b): // returns a list containing one time period:

```
// { 97-06-10 09:00 / 97-06-10 15:00 }
```

shifts(c): // returns a list containing two time periods:

```
// { 97-06-15 00:00 / 97-06-15 08:00,
// 97-06-15 08:00 / 97-06-15 12:00 }
```

### 3.13.6 shifts ( Date\_Range, Time )

shifts creates a list of consecutive one shift time periods which span the given time period. Each time period starts and ends on shift boundaries plus the given time offset, except perhaps the start of the first and end of the last day, which will be the start and end time of the given time period.

Returns: A List of consecutive one day Date\_Ranges with no gaps, spanning the requested time period.

Parameters: - the time period to span (Date\_Range)  
- the length of the shift (Time); must be > 0

Example Usage:

Given...

```
a = date_range("97-06-01 06:00", "97-06-01 15:00");
b = date_range("97-06-10 09:00", "97-06-10 09:18");
c = date_range("97-06-15 00:00", "97-06-16 17:00");
```

Then...

shifts(a, "4 hr"): // returns a list containing three time periods:

```
// { 97-06-01 06:00 / 97-06-01 08:00,
// 97-06-01 08:00 / 97-06-01 12:00,
// 97-06-01 12:00 / 97-06-01 15:00 }
```

shifts(b, "600 s"): // returns a list containing two time periods:

```
// { 97-06-10 09:00 / 97-06-10 09:10,
// 97-06-10 09:10 / 97-06-10 09:18 }
```

shifts(c, "14 hr"): // returns a list containing three time periods:

```
// { 97-06-15 00:00 / 97-06-15 14:00,
// 97-06-15 14:00 / 97-06-16 00:00,
// 97-06-16 00:00 / 97-06-16 14:00 }
// 97-06-16 14:00 / 97-06-16 17:00 }
```

shifts(a, "0 hr"): // returns an error

shifts(a, "-1 hr"): // returns an error

### 3.13.7 shifts ( Date\_Range, Time, Time )

shifts creates a list of consecutive one shift time periods which span the given time period. Each time period starts and ends on shift boundaries plus the given time offset, except perhaps the start of the first and end of the last day, which will be the start and end time of the given time period. The length of a shift is defined by the first time.

Returns: A List of consecutive one day Date\_Ranges with no gaps, spanning the requested time period.

Parameters: - the time period to span (Date\_Range)  
- length of the shift (Time); must be > 0  
- the starting Time of each time period; must be >= 0;  
if greater than the shift length, then first time period  
will span more than a single shift

Example Usage:

Basic Functions	weeks ( Date_Range )
-----------------	----------------------

Given...

```

a = date_range("97-06-01 06:00", "97-06-01 16:00");
b = date_range("97-06-10 09:00", "97-06-10 17:00");
c = date_range("97-06-15 00:00", "97-06-15 17:00");

```

Then...

```

shifts(a, "4 hr", "2 hr"); // returns a list containing three time periods:
// { 97-06-01 06:00 / 97-06-01 10:00,
//   97-06-01 10:00 / 97-06-01 14:00,
//   97-06-01 14:00 / 97-06-01 16:00 }

shifts(b, "8 hr", "600 s"); // returns a list containing two time periods:
// { 97-06-10 09:00 / 97-06-10 16:10,
//   97-06-10 16:10 / 97-06-10 17:00 }

shifts(c, "4 hr", "10 hr"); // returns a list containing three time periods:
// { 97-06-15 00:00 / 97-06-15 10:00,
//   97-06-15 10:00 / 97-06-15 14:00,
//   97-06-15 14:00 / 97-06-15 17:00 }

```

```

shifts(a, "0 hr", "2 hr"); // returns an error

shifts(a, "-1 hr", "0 hr"); // returns an error

```

**3.13.8 weeks ( Date\_Range )**  
weeks creates a list of consecutive one week time periods which span the given time period. Each time period starts and ends on week boundaries (Monday at 00:00), expect perhaps the start of the first and end of the last month, which will be the start and end time of the given time period.

Returns: A List of consecutive one week Date\_Ranges with no gaps, spanning the requested time period.

Parameters: - the time period to span (Date\_Range)

Example Usage:

Given...

```

a = date_range("97-06-04 06:00", "97-06-18 14:00");
b = date_range("97-06-10 02:00", "97-06-14 22:00");
c = date_range("97-06-15 00:00", "97-06-20 00:00");

```

Basic Functions	weeks ( Date_Range, Integer )
-----------------	-------------------------------

Then...

```

weeks(a); // returns a list containing three time periods:
// { 97-06-04 06:00 / 97-06-09 00:00,
//   97-06-09 00:00 / 97-06-16 00:00,
//   97-06-16 00:00 / 97-06-18 14:00 }

weeks(b); // returns a list containing one time period:
// { 97-06-10 02:00 / 97-06-14 22:00 }

weeks(c); // returns a list containing two time periods:
// { 97-06-15 00:00 / 97-06-16 00:00,
//   97-06-16 00:00 / 97-06-20 00:00 }

```

**3.13.9 weeks ( Date\_Range, Integer )**  
weeks creates a list of consecutive one week time periods which span the given time period. Each time period starts and ends on week boundaries plus the given day offset except perhaps the start of the first and end of the last day, which will be the start and end time of the given time period.

Returns: A List of consecutive one week Date\_Ranges with no gaps, spanning the requested time period.

Parameters: - the time period to span (Date\_Range)  
- the day offset (0 = Sunday, 1 = Monday, etc.);  
must be >= 0 and <= 6

Example Usage:

Given...

```

a = date_range("97-06-04 06:00", "97-06-18 14:00");
b = date_range("97-06-07 02:00", "97-06-07 22:00");
c = date_range("97-06-15 00:00", "97-06-20 00:00");

```

Then...

```

weeks(a, 2); // returns a list containing three time periods:
// { 97-06-04 06:00 / 97-06-10 00:00,
//   97-06-10 00:00 / 97-06-17 00:00,
//   97-06-17 00:00 / 97-06-18 14:00 }

weeks(b, 0); // returns a list containing one time period:

```

Basic Functions	Logical Functions
-----------------	-------------------

```
// ( 97-06-07 02:00 / 97-06-07 22:00 )

weeks(c, 4); // returns a list containing two time periods:
// ( 97-06-15 00:00 / 97-06-19 00:00,
// 97-06-19 00:00 / 97-06-20 00:00 )
```

### 3.14 Logical Functions

Logical Functions evaluate logical expressions and return either true or false. They are useful for decision making.

#### 3.14.1 and ( Logical, Logical )

and returns "true" if all of the given logical parameters are "true". If no parameters are given, and returns "true". Evaluation is short circuited. In other words, parameters are evaluated in the order listed. If the evaluation of parameter returns "false", and immediately exits, returning "false" without evaluating the remaining parameters.

Returns: TRUE if all parameters evaluate to "true";  
FALSE otherwise

Parameters: - Any number of Logical expressions.

Example Usage:

```
Given...
a = 5
b = 0
```

```
Then...
and(a > 3, a <= 9) // Returns true
and(a > 3, a <= 9, b == 0) // Returns true
and(a > 6, b == 0) // Returns false
```

#### 3.14.2 not ( Logical )

not computes the negation of its logical parameter. 'not(b)' is equivalent to '!b'.

Returns: TRUE if the parameter evaluates to "false";  
FALSE if the parameter evaluates to "true"

Parameters: - a Logical expression

Example Usage:

Basic Functions	or ( Logical, Logical )
-----------------	-------------------------

```
not(5 > 3) // returns false
not(0 == 1) // returns true
```

#### 3.14.3 or ( Logical, Logical )

or returns "true" if any of the given logical parameters are "true". All parameters are evaluated. If no parameters are given, or returns "false".

Returns: TRUE if any parameter evaluates to "true";  
FALSE if all parameters evaluate to "false"

Parameters: - Any number of logical expressions.

Example Usage:

```
Given...
a = 5
b = 1
```

```
Then...
or(a < 3, a > 9) // Returns false
or(a < 3, a > 9, b != 0) // Returns true
```

#### 3.14.4 xor ( Logical, Logical )

xor computes the sequential exclusive or of all the logical parameters. In other words, xor is applied to the first two parameters. The result becomes the first parameter to the next two value xor evaluation, etc.

The exclusive or of two values is "true" if one of the values is "false" while the other is "true". The result is "false" if both values are "true" or both values are "false".

```
'xor(a, b)' is the same as 'a != b'.
```

Returns: TRUE if the sequential exclusive or of all parameters do not match;  
FALSE otherwise.

Parameters: - Any number of logical expressions.

Example Usage:

```
Given...
a = 5
```

Basic Functions	Miscellaneous Functions
-----------------	-------------------------

b = 1

Then...

xor(a < 3, a > 9) // returns true  
xor(a < 3, a > 9, b != 0) // returns false

3.15 Miscellaneous Functions

3.15.1 id ( Void )

Convert any Model instance to an ID. Strings can be converted too; the format is "0x" followed by a hexadecimal number. For example: id("0x12345678")

Returns: an ID

Parameters: Any Model, or a string

Example Usage:

@~op\_plan.id == op\_plan.id; // is true.  
op\_plan.id.string.id == op\_plan.id; // is true.

ID's can be used to uniquely identify models that don't have unique key-fields (an Operation\_Plan, for example). You can make an ID out of any model, convert it to string (using the string'OIL' function which converts anything to a string), send the string to some 3rd party software package, send it back, convert it back to an ID (using this id(string) OIL function) then search for the model comparing ID objects (which is much faster than comparing strings)

WARNING: Id strings are usually the hex address of the object in memory. Therefore they are only valid for the current invocation of the engine. This is usually good enough for the user-interface uses 'id' was designed for.

3.15.2 current\_index ( Cell )

Returns the current index of a replicating cell

Returns:

Parameters:

Example Usage:

Basic Functions	system ( String )
-----------------	-------------------

3.15.3 system ( String )

Passes 'string' to the operating system for execution (On UNIX, it uses the 'sh' shell).

Returns: If the operating system returns a code from the shell, that code is returned; otherwise 0 is returned. (For UNIX systems, non-zero return values indicate the execution failed or had some sort of error.)

Parameters: The shell command(s) to execute.

Example Usage:

system("if [ -f \$12\_DAT/AM\_backup/sep.i2]; then  
"/bin/mv \$12\_DAT/AM\_backup/sep.i2 \$12\_DAT/AM\_backup/sep.i2.bak; fi");

3.15.4 values ( Type )

If the parameter is an enumeration type, return the list of valid values. (model extension selectors have enumeration types named "<model\_name>\_<selector\_field\_name>\_Enum" Example: values(Operation\_process\_Enum)

Returns:

Parameters:

Example Usage:

3.15.5 data ( Symbol )

This function is obsolete. Find or Create a user-defined DataBase model whose name is given by the parameter string. (see the documentation for the "DataBase" model).

Returns:

Parameters:

Example Usage:

3.15.6 exists ( Void )

Test for nonexistent values. If the parameter returns an error, 'exists' will return the same error. See 'exists\_without\_errors'

Returns: TRUE if it's parameter exists (is not nonexistent).



Basic Functions

exists\_without\_errors ( Void )

Parameters: One parameter of any type.

Example Usage: if (exists(model.field, model.field.string, ""));  
Note: this example would be better coded as: model.field.string ? "",

3.15.7 exists\_without\_errors ( Void )

Test for nonexistent and error values. Throws away any error messages.

Returns: TRUE if it's parameter exists (is not nonexistent), and is not an error.

Parameters: One parameter of any type.

Example Usage: exists\_without\_errors(date("bad date")); // returns FALSE.

3.15.8 export ( string, void )

Writes data to a file or directory of files.

If the first parameter is a directory, and there is a before\_export.in file in the directory, then the OIL commands in that file will be evaluated (see do\_file) before the files in the directory are exported. Similarly if the directory contains an after\_export.in file, then those OIL commands will be evaluated.

The T2\_DATA environment variable is bound to the export directory while evaluating the before\_export.in, after\_export.in and the export worksheets.

Returns: True on success, false on errors.

Parameters: - (pathname [, optional parameters])

The first parameter specifies where to read the export files.

- If it names a directory on the include path, then all the \*.exp files in that directory are read and processed.
- If it names a file on the include path, then that specific file read and processed.

@ ~ The rest of the parameters are passed to the export worksheet.

Example Usage:  
export(directory/file, operation\_plans);

3.15.9 import ( string, void )

Reads data from a file or directory of files.

Basic Functions

engine\_name ( )

If the first parameter is a directory, and there is a before\_import.in file in the directory, then the OIL commands in that file will be evaluated (see do\_file) before the files in the directory are imported. Similarly if the directory contains an after\_import.in file, then those OIL commands will be evaluated.

The T2\_DATA environment variable is bound to the import directory while evaluating the before\_import.in, after\_import.in and the import worksheets.

Returns: True on success, false on errors.

Parameters: - (pathname [, optional parameters])

The first parameter specifies where to read the data files.

- If it names a directory on the include path, then all the \*.imp files in that directory are read and processed.
- If it names a file on the include path, then that specific file read and processed.

The rest of the parameters are passed to the import worksheet.

Example Usage:

import(data/orders, seller, site);

3.15.10 engine\_name ( )

Return the base name of the last saved/opened model file. If there is none, then return the base name of the directory specified by the data option.

Parameters:

Example Usage:

3.15.11 get\_color (String )

Invoke the color chooser.

Returns: the color selected in the color chooser.

Parameters: the color of choice.

Example Usage: get\_color(blue)

3.15.12 get\_drag\_string ( Integer )

Return the nth drag String argument.

Parameters: an integer indicating which drag string argument to return.

Example Usage:

3.15.13 getenv ( String )

Get the value of the operating system environment variable with the specified name String.

Returns: the current setting of the environment variable

Parameters: - an environment variable name

Example Usage: getenv("I2\_IMPORT") //returns the current value of this //environment variable

3.15.14 translate ( String )

Translate the argument to users desired language.

Returns: String

Parameters: String

Example Usage: [A1.title = translate("Promise As Planned");]

3.15.15 model\_file ( )

Return the full path name of the last saved/opened model file.

Parameters: None.

Example Usage:

3.15.16 option ( String )

Returns the value of the named option

Parameters: string name of an option

Example Usage:

3.15.17 setenv ( String, String )

Sets the value of the operating system environment variable named by the first String to the value specified by the second String.

Returns: Returns the value String.

Parameters: - a String (name of variable) - a String (value for variable)

Example Usage:

setenv("I2\_EXPORT", "users/me") //sets the value of //I2\_EXPORT to the directory /users/me

3.15.18 typed\_value ( void )

Convert anything into a Typed\_Value. Typed\_Value is a type/value pair, which allows us to do run-time typing.

If passed a Cell\_State, the typed\_value will be the contents of the worksheet cell pointed to by the Cell\_State.

Returns: a Typed\_Value

Parameters: - One parameter of any type.

Example Usage:

```
@ ~ define(v, typed_value("hi there"));
lv.type: -> String
lv.type == "string"; -> TRUE
lv.value(string): -> "hi there"
lv.string: -> "hi there" // string converts *anything* to a string
```

We can use this to create lists of lists:

```
define(lv_list, list(typed_value(list("one", "two")),
typed_value(list("three", "four", "five"))));
lv_list.first.type: -> list(Typed_Value);
lv_list.for_each(#.value(list(string)) = one, two, three, four, five
```

As a corollary, we can use this to hide lists inside an axis/cross replicator.

Comparing a type to a constant string is very fast, because the

Basic Functions	owner ( Typed_Value )
-----------------	-----------------------

string is automatically converted to a Value\_Type during expression compilation. The comparison is a trivial pointer match. Strings of if's for doing different things with different types should be fast. For example:

```
if(iv.type == "Location", iv.value(Location),
if(iv.type == "Buffer", iv.value(Buffer).location,
if(iv.type == "Resource", iv.value(Resource).location));
```

**3.15.19 owner ( Typed\_Value )**  
Return the owner of a model as a Typed\_Value.

Parameters:

Example Usage:

**3.15.20 first\_value ( Typed\_Value )**  
If the Typed\_Value passed in is a List type, return the first element of the list as a Typed\_Value; otherwise return NONEXISTENT. If the list is empty, return Typed\_Value(element\_type, NONEXISTENT)  
Returns: A Typed\_Value

Parameters: A Typed\_Value

Example Usage:

**3.15.21 delete ( Void )**  
Eliminate the model object passed in the parameter. Use with caution.

Returns: Nothing.

Parameters:

Example Usage:

**3.15.22 make\_type ( Void, Type )**  
make\_type returns a value with a specified type.  
Returns: The first parameter converted to the type specified by the 2nd parameter.

Basic Functions	nonexistent ( )
-----------------	-----------------

Parameters: - a value - a type

Example Usage: normal bb (Buffer\_Plan bp) { variable dd = make\_type(nonexistent, List(Std)); ... {if2 = dd.first}; //won't display until the List has at least one member ... }

**3.15.23 nonexistent ( )**  
Returns the nonexistent value (or lack of a value). This value will cause most other functions to abort.

Parameters:

Example Usage:

**3.15.24 set\_option (String, String)**  
Sets the value of the named option  
Returns: Nothing.  
Parameters:  
- name of a command line option.  
- value for the named command line option.

Example Usage: set\_option("help", "all"); // Prints the list of all options  
**3.15.25 type ( Typed\_Value )**  
Return the type contained in a Typed\_Value.

Parameters: Typed Value.

Example Usage:

**3.15.26 type (String)**  
Converts a string to a Type  
Returns: The named type or MODEL\_ERROR  
Parameters: A string naming the Type to be returned.  
Example Usage: type("integer")

3.15.27 model\_type ( void )

If the parameter is a model instance, return it's Model\_Type.  
If the parameter is a Typed\_Value, and the Typed\_Value contains a model, return the Model\_Type contained by the Typed\_Value.  
Otherwise, return NONEXISTENT.

Returns:

Parameters:

Example Usage:

3.15.28 value ( Typed\_Value, Type )

Return the value member of a Typed\_Value cast to the named type.  
The second argument is a constant string which names the OIL type, like Resource\_Plan. The function checks the correctness of the conversion, and returns NONEXISTENT if it's wrong.  
This function can be used with the model\_choose function to enhance the model\_choose reports.

Parameters:

- a Typed\_Value.

- a constant string which names the OIL type, like Resource\_Plan.

Example Usage:

```
compute chosen_skill = value(typed_value(cell_state), "Resource_plan");  
compute loc = lv.value("Location") ?  
    lv.value("Buffer").location ?  
    lv.value("Resource").location);
```

3.15.29 trace\_time ( )

The trace\_time function can take any number of parameters that return any Type. It will evaluate each of the parameters in order, and return the result of the last one. It also prints how long it took to evaluate the parameters.

Example Usage:

```
$S for_each: user: 244260ms system: 108460ms cpu: 352720ms real 715926ms  
trace_time(  
    supply_chains.find("TOSHIBA").  
    plans.find("TOSHIBA").  
    seller_plans.for_each(#.product_forecasts.
```

```
)  
    for_each(#.entries.for_each(#.accept_as_allocated)))
```

Output:

```
user - CPU milliseconds used to execute the parameters to trace_time  
system - CPU milliseconds used by the system.  
cpu - Total CPU milliseconds (user + system)  
real - Actual (wall-clock) milliseconds (includes time used by other  
processes which are running).
```

Returns: The result from the last parameter.

Parameters: - Any.

3.15.30 memory\_size ( )

Returns the memory actually used by the process which is running the memory\_size function. The memory\_size changes as memory is allocated and freed from the process (see the process\_size function). Therefore memory\_size may decrease even though process\_size stays the same. The difference between the process\_size and the memory\_size is held in reserve for future use. It's typically a megabyte, but could be much more.

Note: this function does not currently work on Microsoft operating systems.

Returns: memory used in bytes.

Parameters: - none.

Example Usages:

```
echo("Memory size: " & string(memory_size / 1024) & "KB");  
echo("Memory size: " & string(memory_size / (1024 * 1024)) & "MB");  
echo("Memory size: " & string(memory_size / (1024 * 1024 * 1024)) & "GB");
```

3.15.31 process\_size ( )

Returns the memory used by the process which is running the process\_size function. The process size will never decrease. The memory allocation sub-system will get large chunks of memory from the system (typically 512KB at a time). It then hands out this memory for various purposes. The 'memory\_size' function will give a more accurate reading of how much memory is actually in use. process\_size is sort-of the max value of memory\_size, plus some overhead.

Basic Functions	with_connection ( Connection, void )
-----------------	--------------------------------------

Note: Many system commands report memory usage in "kilobytes" by taking total memory size and dividing by 1024 (NOT 1000). If the results from process\_size do not match what's reported by your favorite system command, this could be the cause. Note: some system commands (e.g. ps) do not include shared memory, or memory that's swapped out.

Note: this function does not currently work on Microsoft operating systems.

Returns: process size in bytes.

Parameters: - none.

Example Usages:  
echo("Process size: " & string(process\_size / 1024) & "KB");  
echo("Process size: " & string(process\_size / (1024 \* 1024)) & "MB");  
echo("Process size: " & string(process\_size / (1024 \* 1024 \* 1024)) & "GB");

### 3.15.32 with\_connection ( Connection, void )

Like 'do' except the first parameter is a Connection. All other arguments are executed in the context (with a Cell\_State) of the Application\_Report for the connection.

Note: Many system commands report memory usage in "kilobytes" by taking total memory size and dividing by 1024 (NOT 1000).

Returns: The result from the last parameter.

Parameters: - A Connection model.

Example Usage:  
with\_connection(users.find("somebody").active\_ui,  
display\_report("report\_name", parameters));

### 3.15.33 functions ( Symbol )

Get the list of Functions whose name matches the parameter string. Since there can be lots of functions with the same name (but different parameters), the Function model can't have a 'find' method. This function is used instead. This function is \*much\* faster than filtering the list of all functions.

Returns: The list of Functions whose name matches the parameter string.

Parameters: a function name

Basic Functions	echo ( Logical )
-----------------	------------------

Example Usage:  
// Get the documentation for the 'enclose' function that takes a 'Date\_Range' parameter  
functions("enclose").filter(#argument\_types.first == "Date\_Range").first.documentation;

### 3.15.34 echo ( Logical )

In the do\_file function, expressions are echoed by default. echo(false); will turn off echoing. echo(true); will turn it back on.

### 3.15.35 echo ( String )

Print a constant string to "stderr".

Returns: Nothing.

Parameters:

- a string to print to stderr.

Example Usage:

### 3.15.36 with\_echo\_file ( String, void )

Similar to 'do', except the first parameter is a filename. All the other parameters will be evaluated with their output going to the specified file.

Returns: The value of the last parameter

Parameters: - The first parameter is a filename, the other parameters can be anything.

Example Usage:  
with\_echo\_file("filename", echo("Hello World"));

### 3.16 File Functions

File functions

### 3.16.1 is\_directory ( String )

Returns true if the first parameter is the pathname to a directory.

Returns: True if the first parameter is the pathname to a directory.

Parameters: - A pathname string

Basic Functions	is_file (String)
-----------------	------------------

Example Usage:  
if (is\_directory("something"), import("something"));

### 3.16.2 is\_file (String)

Returns true if the first parameter is an existing filename.

Returns: True if the first parameter is an existing filename.

Parameters: - A pathname string

Example Usage:  
if (is\_file("something"), import("something"));

### 3.16.3 is\_writable (String)

Returns true if the first parameter is writable file or directory.

Returns: True if the first parameter is writable file or directory.

Parameters: - A pathname string

Example Usage:  
if (is\_writable("something"), export("something"));

## 3.17 Evaluation Functions

Functions for controlling the evaluation of Expressions.

### 3.17.1 background (Expression)

The 'background' function can take any number of parameters that return any Type. It will evaluate each of the parameters in order, and return the result of the last one. The parameter expressions may be executed in the background (some of the planning functions will do this). No reply will be sent back to the UI.

Returns: The result from the last expression.

Parameters: - Any

Example Usage:  
plan\_site\_plans\_for\_each(#,promise\_as\_planned).background;  
background(active\_strategy.run, wait, update\_report);

Basic Functions	do ( )
-----------------	--------

3.17.2 do ( )  
The 'do' function can take any number of parameters that return any Type. It will evaluate each of the parameters in order, and return the result of the last one. One of the parameters can be a 'define' function, which defines variables local to the 'do'.

Returns: The last parameter.

Parameters: - Any

Example Usage:  
do(define(hi, "Greetings"), echo(hi), echo("From " & user.name));

### 3.17.3 call (Symbol)

Evaluate the function worksheet specified by the first parameter, passing parameters specified by the rest of the parameters.

Parameters:

The first parameter is the name of the function worksheet. This must be string constant or identifier. The current user's report path is searched for a file with this name and a ".fws" (functional worksheet) extension.

The rest of the parameters are passed to the function worksheet. If a variable is passed as a parameter, the called function worksheet can change the value of the variable by setting the parameter (e.g. the variable is "passed by reference").

Returns:

If the named function worksheet has a cell named 'return', then the contents of that cell are returned; otherwise 'call' returns void.

Example Usage:  
define(result2, make\_type(nonexistent, string);  
define(result, call(my\_oil\_function, result2, 123));

### 3.17.4 do\_file\_echo (String)

Evaluates all of the expressions in the specified file. The expression and the results are sent to stdout, unless the first character in the file is '@', or 'echo(of)'; is used. (reminiscent of old msdos batch files...)

Parameters: The name of the file to read OIL expressions from. If the file name is "- ", OIL commands are read from stdin. This can be a useful debugging tool.

Basic Functions	do_file ( String )
-----------------	--------------------

Example Usage: do\_file("\$I2\_HOME/custom/my\_commands.in");

3.17.5 do\_file ( String )

Like 'do\_file\_echo', except it will optimize itself into a simple 'do' if the filename parameter is a string constant, and "echo" is turned off at the time 'do\_file' is compiled. That is, if it's compiled inside another 'do\_file' where 'echo(off)' is used, or it's invoked as '@do\_file("some\_file.in")'

Warning: 'do\_file' will be changed to never echo statements in 3.07. Change your scripts to use 'do\_file\_echo' when you want echoing (typically only regression tests need echoing, and all other production uses don't).

Returns:

If in-lined, The results from the last expression in the file.  
Otherwise, it returns void (nothing).

Parameters: The name of the file to read OIL expressions from. If the file name is "-", 'do\_file\_echo' is invoked to read OIL commands from stdin. This can be a useful debugging tool.

Example Usage: do\_file("\$I2\_HOME/custom/my\_commands.in");

3.17.6 do\_file\_fast ( String )

Like 'do\_file\_echo', except it will optimize itself into a simple 'do'. The filename parameter must be string constant.

Warning: This function is obsolete in 3.06 (replaced by 'do\_file').

Parameters: The name of the file to read OIL expressions from.

Returns: The results from the last expression in the file.

Example Usage: do\_file\_fast("\$I2\_HOME/custom/my\_commands.in");

3.17.7 engine ( Expression )

Send the expression to the engine for evaluation. This is primarily used in Command and Control Expressions that would normally execute in the UI or API clients.

Parameters:

- expression that is to be sent to the engine for evaluation.

Basic Functions	evaluate ( Expression )
-----------------	-------------------------

Example Usage:

3.17.8 evaluate ( Expression )

Compiles and executes the OIL expression passed as a parameter. The expression will be evaluated in the current context, so it can use any local variables, cells, etc.

Parameters: The string to evaluate

Returns: TRUE if there were no compile error, else FALSE

Example Usage: evaluate(model.user\_field); // user\_field is a string.

3.17.9 if ( Logical, Void, Void )

The 'if' function takes two or three parameters. The first must return Logical. If the first returns "true", then the second parameter is evaluated and its result returned; otherwise, the third parameter (when present) is evaluated and its result returned. Note that if there are 3 parameters, the second and third must return the same Type, and that will be the Type returned by the 'if' function. If there are two parameters, 'if' returns the "Void" type.

Note that only one of the second or third parameters is evaluated; the other is not evaluated at all. This means that the Logical could be checking for an error condition in one of the other parameters so that the error can be avoided. For example, if(denom != 0, num/denom, 0)' prevents division by zero errors by checking the denominator and not performing the division if it is 0.

Parameters:

Example Usage:

3.17.10 while ( Logical, Void )

Looping function; loop while the first parameter returns true.

WARNING: Most 'while' loops can be coded much more efficiently using the 'for\_each' function. Any 'while' loop that appends to a list should probably be re-coded to use 'for\_each'.

Parameters:

The first parameter is an expression that returns FALSE when the loop should exit. while(true) loops forever until the number of iterations specified by the 'while\_max' option is exceeded (the default is 4,000,000). The rest of the parameters are simply executed.

Returns: nothing.

Example Usage:

```
define(count, 3);
while(count > 0, echo(count,string), define(count, count - 1));
// Note: it would be much better to use:
// integers(1, 3).for_each(#,string,echo);
```

### 3.17.11 reference ( Variable )

Returns a Reference to a variable. A reference can be used with the 'typed\_value' function to get the current value of the variable, and with the 'set\_cell' function to set the variable.

Returns: A Reference to the variable

Parameters: A variable

Example Usage:

```
variable foo = "Howdy"; // type is 'String'
variable foo_reference = reference(foo); // type is 'Reference'
foo_reference.typed_value(string ""); // equals "Howdy"
set_cell(foo_reference, typed_value("Hi")); // Sets 'foo' to "Hi"
```

## 3.18 RhythmLink Functions

Functions relating to RhythmLink

### 3.18.1 rl\_field ( Data\_Table, String )

This function is used in a rhythmLink import layout. The current record is controlled by the traversal mechanism of the import layout.

Returns: current field value belonging to a specific field from the current record of a table.

Parameters:

- first parameter represents the table.
- second parameter represents the field name.

Example Usage:

### 3.18.2 rhythmLink\_field ( Symbol, String )

This function returns the named field of a record. It is used internally by a Data\_Copy\_Map.

Returns: a Value Type

Parameters: - transaction id, field name

Example Usage:

### 3.18.3 rl\_record\_field ( Record, String )

This function returns the named field of a record.

Returns: a Value Type

Parameters:

- record
- field name

Example Usage:

## 3.19 Debug Functions

### Functions for OIL debugging

A simple lly debugger is provided with the following OIL functions (optional parameters are in square [brackets])

```
@ ~ break([expression [, name [, condition]])
breakpoint([, name])
stop([cell_name [, worksheet_name])
enable([breakpoint_name])
disable([breakpoint_name])
next([count])
step([count])
cont([count])
watch([expression [, name [, condition]])
unwatch([watchpoint_name])
where
whereis
print_variables
```



Basic Functions	stop ( Symbol, Symbol )
-----------------	-------------------------

print\_report\_variables  
inspect(any\_model)

When stopped at a breakpoint, an "OIL listener" will be entered, which reads from scp\_engine's standard-in, and writes to standard-out. You will see a "ODB>" prompt (Oil Debugger). You can enter any OIL expression, and see the evaluation results. Entering an empty line will cause the previous command to be run again. This is handy for running 'next' step or 'cont' repeatedly.

Note: The debugger prompt can be customized via the 'debug\_prompt' option. For example:

set\_option("debug\_prompt", "Break (0)>")  
will use the breakpoint name in the prompt (e.g. "Break 3>")

Warning: This means that you can't run scp\_engine in the background and still use these debugging aids!

Suggestion: Run scp\_engine in an emacs shell buffer. That way you can use meta-p to recall previous commands, you get infinite history, and it's easy to kill&yank (or cut&paste).

NOTE: All breakpoints default to disabled if the 'debug' option is FALSE (that's the default!). Breakpoints default to enabled when the debug option is TRUE. Breakpoint's can be controlled through the Breakpoint model. See the 'enable' and 'disable' OIL functions.

3.19.1 stop ( Symbol, Symbol )

Creates a Breakpoint for the named cell and worksheet. See the 'break' function for more details.

3.19.2 stop ( Symbol )

Creates a Breakpoint for the named cell in the current worksheet. See the 'break' function for more details.

3.19.3 break ( Expression, Symbol, Expression )

Executes the first parameter, returning it's value. Also creates a Breakpoint model for the expression.

Parameters:

If no parameters are specified, the current list of breakpoints is printed.

1st parm can be of any type, and is the result from this function.

Basic Functions	breakpoint ( )
-----------------	----------------

2nd parm is the optional breakpoint name (the default is a small integer). Note: There can be multiple breakpoints with the same name. All will be enabled & disabled together, have the same condition, etc.

3rd parm is an optional breakpoint condition which must evaluate to TRUE before breaking. 'condition' is evaluated before 'expression'.

3.19.4 breakpoint ( )

Create a breakpoint

3.19.5 breakpoint ( Symbol )

Find or create a breakpoint named 'name'

3.19.6 disable ( )

Disable all breakpoints

3.19.7 disable ( Symbol )

Disable the named breakpoint If 'name' is "all", then all breakpoints are disabled.

3.19.8 enable ( )

Enable all breakpoints

3.19.9 enable ( Symbol )

Enable the named breakpoint If 'name' is "all", then all breakpoints are enabled.

3.19.10 next ( )

Run until the next cell, do\_file line or variable assignment. Does not cross into other 'call' or 'do\_file' files (see step).

3.19.11 step ( )

Run until the next cell, do\_file line or variable assignment. Will step into 'call' or 'do\_file' statements (see next)

3.19.12 cont ( )

Continue until the next breakpoint.

3.19.13 next ( Integer )

Run until the next cell, do\_file line or variable assignment. Does not cross into other 'call' or 'do\_file' files (see step). 'count' is the number of breakpoints to skip before actually stopping.

3.19.14 step ( Integer )

Run until the next cell, do\_file line or variable assignment. Will step into 'call' or 'do\_file' statements (see next) 'count' is the number of breakpoints to skip before actually stopping.

3.19.15 cont ( Integer )

Skip 'count' minus one breakpoints, then stop.

3.19.16 watch ( Expression, Symbol, Expression )

Creates a 'watchpoint' model object that will execute 'expression' and print the results at every breakpoint executed in the same context as 'watch' was executed in (the same worksheet or do\_file).

Parameters:

If no parameters are specified, the current list of watchpoints is printed.

1st parm can be of any type, and is the result from this function.

2nd parm is the optional watchpoint name (the default is a small integer). Note: If there is already a watchpoint with the given name, it's expression and condition are replaced with new definitions.

3rd parm is an optional watchpoint condition which must evaluate to TRUE before printing. 'condition' is evaluated before 'stuff'.

If the 3rd 'condition' parameter is 'nonexistent' (the default) the watchpoint will print only when the results are different then the last time. To print every time, use TRUE for the condition.

3.19.17 unwatch ( Symbol )

Removes the named 'watchpoint' model object (created by 'watchpoint')

Parameters: The name of the watchpoint to delete. If the name is 'all', then all watchpoints are deleted.

3.19.18 where ( )

Return a one-line description of where this function is executing.

3.19.19 whereis ( )

Print a detailed description of where this function is executing

3.19.20 print\_variables ( )

3.19.21 print\_report\_variables ( )

Print all the report variables and their values.

3.19.22 inspect ( void )

Inspect any model, displaying all it's fields and sub-models. You can "drill down" into the model using the inspectS, inspectH and inspectHS functions.

Parameters: Any model, or list(model)

Example: supply\_chains.inspect; // results in:

1 = Inspection of Supply\_Chain:

==> name schain

==> description A simple supply chain schain

0 ==> sites A-SUPL, A-LINK

1 ==> top\_sites A-LINK, A-SUPL

2 ==> sellers LINK-SEL, LINK-SUB-SEL2, LINK-SUB-SEL1, SUPL-

SEL

3 ==> top\_sellers SUPL-SEL, LINK-SEL

4 ==> plans plan1

3.19.23 inspectS ( Integer )

Drill down into a field displayed by the previous 'inspect\*' call.

Parameters: 'inspect' numbers the fields it prints. Pass one of these numbers into inspectS.

3.19.24 inspectH ( Integer )

Recall a previously inspected value.

Parameters: Each time one of the 'inspect\*' functions is called, the first line shows a history number. Using that number as the parameter to inspectH prints the same inspection again.

3.19.25 inspectHS ( Integer, Integer )

Drill down into a field of a previously inspected value.

Parameters:

History\_number: Each time one of the 'inspect\*' functions is called, the first line shows a history number.

@ ~ field\_number: 'inspect' numbers the fields it prints. Pass one of these numbers into inspectHS.

3.19.26 inspectV (Integer)

Return the value of a field from the last inspected model (or list).

Parameters: inspect' numbers the fields it prints. Pass one of these numbers into inspectV.

3.19.27 inspectV (Integer, Integer)

Return the value of a field from the last inspected model (or list).

Parameters:  
History\_number: Each time one of the 'inspect\*' functions is called, the first line shows a history number.

@ ~ field\_number: 'inspect' numbers the fields it prints. Pass one of these numbers into inspectV

3.20 Program Functions

Functions for program control

3.20.1 user ( )

Returns the User model for the client running this function. If you are executing an OIL expression where there is not a user (for example, the 'startup' expression), then the 'user' OIL function returns the model for the 'unspecified' user.

Parameters: None.

Example Usage:

3.20.2 quit (String)

Quit this client; notify the engine accordingly. The parameter String explains why.

Returns: Nothing.

Parameters:  
- string explaining why this client is quitting.

Example Usage:

3.20.3 shutdown (String)

Ask engine to shutdown. The parameter String explains why.

Returns: Nothing.

Parameters:  
- string explaining why the engine is being shut down.

Example Usage:

3.21 Engine\_Queue Functions

Engine\_Queue Commands

3.21.1 wait ( )

Cause future engine commands to wait until all previous commands have finished.

'wait' only affects the engine<->gui communications mechanisms. You can use it (only) on the gui to serialize engine requests with different priorities.

In general, there are two classes of commands, those that only display data (e.g. display\_report, update\_report), and those that modify data (e.g. setting a field in a model). The commands that only display data have a higher-priority than the modify commands, so they normally run first. 'wait' can be used to change that.

For example, you may have an action statement that sets a field, then displays a report. Setting a field has a lower priority than displaying a report, so the set will normally happen after the display. 'wait' can be used between the set and display\_report to force the display\_report to wait until the set has finished. 'wait' should be needed rarely (if at all, see dis::7433).

Note: the various planning actions start up "background" tasks on the engine. 'wait' does not wait for these to complete (it could hang-up your gui for hours...). (some people want 'wait' to work for these anyway, see dis::9189)

Returns: Nothing.

Parameters: - None.

Example Usage:  
do(foo.set\_some\_value(123),  
wait,  
display\_report("some\_report", foo));

## 4 Summary Section

### Models

#### Model Supply\_Chain

The Fields in this Model are  
name, description, sites, top\_sites, sellers, top\_sellers, plans.

#### Model Site

The Fields in this Model are  
name, description, category, sub\_category, rank, organization, members, member\_of (Explicit\_Site), role, margin\_target (Date, Date), delivery\_name, delivery\_contact, delivery\_phone, delivery\_fax, delivery\_address, delivery\_city, delivery\_state, delivery\_country, delivery\_postal\_code, billing\_name, billing\_contact, billing\_phone, billing\_fax, billing\_address, billing\_city, billing\_state, billing\_country, billing\_postal\_code, tax\_identification, site\_plan (Plan), owner.

#### Model Seller

The Fields in this Model are  
name, description, category, sub\_category, rank, organization, members, member\_of (Seller), site\_groups, all\_site\_groups, site\_group (Symbol), product\_roots, products, top\_products, active\_products, product\_groups, top\_product\_groups, forecast\_horizon, first\_day\_of\_week, week\_periods, request\_naming, atp\_horizon, seller\_plan (Plan), owner.

#### Model Plan

The Fields in this Model are  
name, description, current, horizon, default\_operation\_plan\_rank, site\_plans, top\_site\_plans, seller\_plans, top\_seller\_plans, the\_problems, problems, problems (Date\_Range), problems (Problem\_Category), problems (Date\_Range, Problem\_Category), problem\_categories, top\_active\_strategies, active\_strategies, auto\_run\_strategy, background\_run\_strategy, resource\_plan (Resource), buffer\_plan (Buffer), operation\_plans (Operation), forecast (Product), forecast (Product\_Root), run\_for\_now (integer), owner.

#### Model Problem\_Category

The Fields in this Model are  
name, short\_name, full\_name, description, remark, super\_categories, sub\_categories, lead\_categories, fields.

#### Model Operation

The Fields in this Model are  
name, description, category, sub\_category, interruptible, loads, flows, consume\_flows, produce\_flows, all\_consume\_flows, all\_produce\_flows, all\_flows, sub\_operations, super\_operations, process, problem\_detectors, planning\_buffers, planning\_resources, operation\_plans (Plan), unit, release\_fence, release\_fence\_date (Plan), release\_soon\_fence, release\_soon\_fence\_date (Plan), release\_name\_expression, next\_release\_number, release\_number, owner.

#### Model Operation\_Plan

The Fields in this Model are  
operation, remark, rank, released, release\_name, release\_fence\_date, release\_soon\_fence\_date, use\_alternate (Operation\_Plan), use\_alternate (Operation\_Plan, Operation), motive, units, quantity, std\_time, expedite, dates, hint, locked\_as\_planned, load\_plans, flow\_plans, all\_consuming\_flow\_plans, all\_producing\_flow\_plans, sub\_operation\_plans, super\_operation\_plan, top\_operation\_plan, process, problems, problems (Date\_Range), problems (Problem\_Category), problems (Date\_Range, Problem\_Category), problem\_categories, split (Item, Quantity, Logical), split (Quantity), Restriction), owner.

#### Model Resource

The Fields in this Model are  
name, description, category, sub\_category, location, skills, efficiency, variability, maintenance, size, load\_policy, problem\_detectors, resource\_plan (Plan), owner.

#### Model Resource\_Plan

The Fields in this Model are  
resource, remark, efficiency\_profile (Date\_Range), efficiency\_average (Date\_Range), efficiency\_average (Date), efficiency\_profile\_at\_skill (Skill, Date\_Range), efficiency\_average\_at\_skill (Skill, Date\_Range), efficiency\_average\_at\_skill (Skill, Date), available\_time (Date\_Range), capacity (Date\_Range), efficiency, load\_plans, load\_plans (Date\_Range), load\_time (Date\_Range), load\_std\_time (Date\_Range), load\_policy, balance\_time (Date\_Range, Logical, Logical), balance\_std\_time (Date\_Range, Logical, Logical, Logical), move (Load\_Plan, Logical, Logical), previous\_gap (Load\_Plan), next\_gap (Load\_Plan), size, size\_profile (Date\_Range), size\_usage\_profile (Date\_Range), available\_sized\_time (Date\_Range), sized\_capacity (Date\_Range), load\_sized\_time (Date\_Range), load\_sized\_std\_time (Date\_Range), balance\_sized\_time (Date\_Range, Logical, Logical), balance\_sized\_std\_time

Summary Section	wait ( )
-----------------	----------

(Date\_Range, Logical, Logical, Logical), maintenance, problems, problems (Date\_Range), problems (Problem\_Category), problems (Date\_Range, Problem\_Category), problem\_categories, problem\_time (Date\_Range), problem\_std\_time (Date\_Range), problem\_sized\_time (Date\_Range), problem\_sized\_std\_time (Date\_Range), resolve (Problem, Logical, Logical, Logical), resolve (Date\_Range, Logical, Logical, Logical), owner.

#### Model Buffer

The Fields in this Model are

name, description, category, item, location, flow\_policy, stocking\_policy, problem\_detectors, level, all\_producing\_operations, all\_consuming\_operations, unit, buffer\_plan (Plan), all\_supplying\_operations, owner.

#### Model Buffer\_Plan

The Fields in this Model are

buffer, remark, flow\_plans, flow\_plans (Date\_Range), flow\_plans (Date\_Range, Logical), producing\_flow\_plans, producing\_flow\_plans (Date\_Range), producing\_flow\_plans (Date\_Range, Logical), consuming\_flow\_plans, consuming\_flow\_plans (Date\_Range), producing\_flow (Date\_Range), consuming\_flow (Date\_Range), on\_hand\_profile, on\_hand\_profile (Date\_Range), on\_hand, on\_hand (Date), on\_hand (Date\_Range), on\_hand (Date\_Range, Logical), on\_hand (Flow\_Plan), on\_hand\_date, on\_handLots (Date), lots, flow\_policy, create\_producing\_operation\_plan (Measure\_Unit, Restriction), create\_consuming\_operation\_plan (Measure\_Unit, Restriction), problems (Date\_Range), problems (Problem\_Category), problems (Date\_Range, Problem\_Category), problem\_categories, in\_flow\_plans, in\_flow\_plans (Date\_Range), in\_flow\_plans (Date\_Range, Logical), out\_flow\_plans, out\_flow\_plans (Date\_Range), out\_flow\_plans (Date\_Range, Logical), in\_flow (Date\_Range), out\_flow (Date\_Range), owner.

#### Model Product\_Root

The Fields in this Model are

name, description, suppliers, min\_quantity, min\_delivery\_lead\_time, delivery\_area, effective\_dates, customer, customers, forecast\_policy, product, product (Product\_Allocation), unit, owner.

#### Model Product

The Fields in this Model are

Summary Section	wait ( )
-----------------	----------

product\_root, name, rank, description, items, min\_quantity, min\_delivery\_lead\_time, delivery\_area, customer, customers, forecast\_policy, active, expiration\_policy, allocation\_policy, auto\_allocate, always\_override\_members\_forecasted, always\_override\_members\_committed, availability\_policy, price\_policy, consume\_earlier, time\_pad, quantity\_pad, quote\_end\_of\_bucket, quote\_multiple, quote\_larger\_multiple, allocate\_quote\_multiple, auto\_allocate\_from\_organization, generic\_products, specific\_products, active\_specific\_products, alternate\_products, primary\_products, active\_primary\_products, groups, top, owner.

#### Model Forecast

The Fields in this Model are

level, name, remark, active, root, member\_forecasts, active\_member\_forecasts, generic\_forecasts, active\_generic\_forecasts, specific\_forecasts, active\_specific\_forecasts, request, entries, entries\_consumed, entries\_members\_consumed, accept\_as\_allocated, allocate\_allocated\_available, owner.

#### Model Site\_Plan

The Fields in this Model are

site, description, organization\_plan, members, role, owner.

#### Model Seller\_Plan

The Fields in this Model are

seller, organization, members, forecasts, active\_forecasts, forecast (Symbol), top\_forecasts, active\_top\_forecasts, product\_root\_forecasts, active\_product\_root\_forecasts, product\_forecast (Symbol), forecast\_horizon, atp\_horizon, actual\_requests, actual\_promises, problems, problems (Date\_Range), problems (Problem\_Category), problems (Date\_Range, Problem\_Category), accept\_as\_allocated, owner.

#### Model Problem

The Fields in this Model are

description, dates, feasible, cost, last\_change, category, in\_category (Problem\_Category), interaction, resolvable, resolve, resolve (Active\_Strategy), owner.

#### Model Active\_Strategy

The Fields in this Model are

Summary Section	wait()
-----------------	--------

strategy, super, remark, date\_activated, reset\_date,activated, run, stop, continue, running, stopped, run\_time, stable\_time, target\_achieved, interaction, annealing\_goodness, resolve\_count, permanent, auto\_run, background\_run, active\_problems, problems (Date\_Range), problems (Problem\_Category), problems (Date\_Range, Problem\_Category), problem\_categories, active\_goals, termination, execution, problem\_selection, owner.

#### Model Location

The Fields in this Model are  
name, description, category, sub\_category, super\_location, sub\_locations, within (Location), box, x\_position, x\_size\_min, x\_size, y\_position, y\_size\_min, y\_size, owner.

#### Model Item

The Fields in this Model are  
name, description, category, drawing\_id, family, children, artificial, spec, lots\_tracked, delivery, buffers, buffer\_plans (Plan), unit, owner.

#### Model Skill

The Fields in this Model are  
name, description, resources, selection, loading\_operations, loading\_buffers, owner.

#### Model Configuration

The Fields in this Model are  
item, spec, interchangeable (Configuration), owner.

#### Model Item\_Group

The Fields in this Model are  
name, description, top, root, sub\_groups, sub\_items, all\_items (List(Item)), owner.

#### Model Operation\_State

The Fields in this Model are  
operation\_plan, unattach, operation\_plans, identifier, date, state\_spec, owner.

#### Model Request

The Fields in this Model are

Summary Section	wait()
-----------------	--------

name, description, delivery\_requests, delivery\_policy, delivery\_naming, accept\_by, promise, promise\_by, date\_issued, date\_accepted, date\_queued, cancel, plan\_to\_satisfy, plan\_as\_requested, seller\_plan, forecast, customer\_plan, name\_from\_customer, delivery\_name, delivery\_contact, delivery\_phone, delivery\_fax, delivery\_address, delivery\_city, delivery\_state, delivery\_country, delivery\_postal\_code, last\_change, issued, accepted, queued, problems, problems (Date\_Range), problems (Problem\_Category), problems (Date\_Range, Problem\_Category), owner.

#### Model Promise

The Fields in this Model are  
request, delivery\_promises, delivery\_policy, acceptance, accept\_by, date\_offered, plan\_to\_satisfy, promise\_as\_planned, promise\_as\_planned (Logical), plan\_as\_promised, reject, last\_change, problems, problems (Date\_Range), problems (Problem\_Category), problems (Date\_Range, Problem\_Category), offered, owner.

#### Model Acceptance

The Fields in this Model are  
promise, delivery\_acceptances, accepted, accept\_by, plan\_to\_satisfy, plan\_as\_accepted, accept\_as\_promised, accept\_as\_promised (Logical), last\_change, problems, problems (Date\_Range), problems (Problem\_Category), problems (Date\_Range, Problem\_Category), owner.

#### Model Horizon

The Fields in this Model are  
name, description, bucket\_spec, buckets (Date\_Range).

#### Model Horizon\_Bucket\_Start

The Fields in this Model are  
start, owner.

#### Model Site\_Group

The Fields in this Model are  
name, description, sites, sites (List(Site)), spec, explicit\_sites, owner.

#### Model Product\_Group

The Fields in this Model are  
name, description, top, root, use\_std\_split, sub\_groups, sub\_products, all\_products, all\_products (List(Product)), all\_products\_and\_specifics, unit, owner.

Summary Section	wait()
-----------------	--------

### Model Delivery\_Request

The Fields in this Model are

name, actual, forecast\_entry, delivery\_promise, item\_requests, due, promising\_policy, fulfillment\_policy, max\_price, seller\_plan, customer\_plan, name\_from\_customer, rank, delivery\_name, delivery\_contact, delivery\_phone, delivery\_fax, delivery\_address, delivery\_city, delivery\_state, delivery\_country, delivery\_postal\_code, promised\_late, promised\_early, promised\_overpriced, cancel, plan\_to\_satisfy, plan\_as\_requested, not\_planned, planned\_late, planned\_early, last\_change, problems (Date\_Range), problems (Problem\_Category), promised\_date\_problem, promised\_price\_problem, plan\_problems, owner.

### Model Delivery\_Promise

The Fields in this Model are

delivery\_request, delivery\_acceptance, delivery\_atp, promising\_policy\_atp, item\_promises, due, rank, fulfillment\_policy, date\_confirmed, list\_price, sum\_discount, sum\_price, delivery\_discount, delivery\_price, promised\_late, promised\_early, promised\_overpriced, plan\_to\_satisfy, plan\_as\_promised, promise\_as\_planned, promise\_by\_policy, not\_planned, planned\_late, planned\_early, last\_change, problems (Date\_Range), problems (Problem\_Category), problems (Date\_Range, Problem\_Category), confirmed, promised\_date\_problem, promised\_price\_problem, plan\_problems, owner.

### Model Delivery\_Acceptance

The Fields in this Model are

delivery\_promise, item\_acceptances, due, fulfillment\_policy, plan\_to\_satisfy, plan\_as\_accepted, accept\_as\_promised, not\_planned, planned\_late, planned\_early, last\_change, problems (Date\_Range), problems (Problem\_Category), problems (Date\_Range, Problem\_Category), plan\_problems, delivery\_not\_coordinated\_problem, owner.

### Model Strategy

The Fields in this Model are

name, description, deterministic\_resolvers, deterministic\_problem\_selection, problem\_sets, changes, locks, default\_change\_focus, goals, termination, execution, problem\_selection, max\_stable\_time, max\_stable\_resolve\_count, max\_run\_time, max\_resolve\_count, max\_heat, min\_heat, run (Plan), do\_before, do\_after, do\_before\_resolve.

### Model Problem\_Set

The Fields in this Model are

Summary Section	wait()
-----------------	--------

fence, min\_time, last\_change\_after\_activated, category, min\_cost, must\_resolve, focus, feasible\_focus, horizon, owner.

### Model Strategy\_Change

The Fields in this Model are

change\_category, focus, owner.

### Model Strategy\_Lock

The Fields in this Model are

name, spec, owner.

### Model Strategy\_Goal

The Fields in this Model are

goal, focus\_to\_target, focus\_beyond\_target, owner.

### Model Active\_Problem

The Fields in this Model are

problem, focus, unresolvable, interaction, must\_resolve, owner.

### Model Active\_Goal

The Fields in this Model are

strategy\_goal, goal, adjusted\_value, adjusted\_target, focus\_value, owner.

### Model Explicit\_Site

The Fields in this Model are

site, owner.

### Model Unit

The Fields in this Model are

preferred\_measure, discrete, quantities, convert (Measure\_Unit, Measure), convert (Measure\_Unit, Measure\_Unit).

### Model Unit\_Quantity

The Fields in this Model are

quantity, owner.

### Model Product\_Supplier

The Fields in this Model are

supplier, items, owner.

### Model Product\_Item

Summary Section	wait ( )
-----------------	----------

The fields in this Model are  
item, owner.

**Model Generic\_Product**  
The fields in this Model are  
product, quantity\_per, owner.

**Model Alternate\_Product**  
The fields in this Model are  
product, quantity\_per, rank, owner.

**Model Sub\_Product\_Group**  
The fields in this Model are  
product\_group, quantity\_per, std\_split, root, owner.

**Model Sub\_Product**  
The fields in this Model are  
product, quantity\_per, std\_split, root, owner.

**Model Forecast\_Entry**  
The fields in this Model are  
delivery\_dates, date\_forecasted, forecasted, cumulative\_forecasted, specifics\_forecasted, members\_forecasted, override\_members\_forecasted, date\_committed, committed, cumulative\_committed, specifics\_committed, members\_committed, override\_members\_committed, retain\_from\_allocated, lock\_retain\_from\_allocated, retain\_from\_accepted, planned, cumulative\_planned, specifics\_planned, members\_planned, date\_allocated, allocated, lock\_allocated, cumulative\_allocated, specifics\_allocated, members\_allocated, date\_accepted, accepted, cumulative\_accepted, members\_consumed, accept\_as\_allocated, consumed, cumulative\_consumed, specifics\_consumed, members\_consumed, actual\_promises, available\_to\_promise, available, cumulative\_available, members\_available, specifics\_available, allocated\_available, cumulative\_allocated\_available, planned\_available, cumulative\_planned\_available, zero\_available\_to\_promise, allocate\_allocated\_available, forecast\_policy, allocation\_policy, forecast\_requests, forecast\_promises, owner.

**Model ATP\_Entry**  
The fields in this Model are  
available\_dates, allocated, consumed, allocated\_available, cumulative\_allocated\_available, owner.

Summary Section	wait ( )
-----------------	----------

**Model Item\_Request**  
The fields in this Model are  
name, configuration, item, quantity, cancel, cancelled, item\_promise, max\_price, promised\_short, promised\_excess, promised\_overpriced, delivery\_plan, plan\_to\_satisfy, receiving\_plan, plan\_as\_requested, not\_planned, planned\_late, planned\_early, planned\_short, planned\_excess, last\_change, problems, problems(Date\_Range), problems(Problem\_Category), problems(Date\_Range, Problem\_Category), promised\_quantity\_problem, promised\_price\_problem, planned\_date\_problem, planned\_quantity\_problem, owner.

**Model Load**  
The fields in this Model are  
name, usage\_policy, resource, skill, load\_sizes, start\_setup, end\_setup, start\_location, end\_location, owner.

**Model Flow**  
The fields in this Model are  
name, buffer, phantom, produced, usage\_policy, owner.

**Model Operation\_Problem\_Detector**  
The fields in this Model are  
detector, feasible, cost, owner.

**Model Load\_Plan**  
The fields in this Model are  
load, resource\_plan, dates, hint, hint\_on, lock\_on, use\_alternate, use\_alternate(Resource\_Plan), owner.

**Model Flow\_Plan**  
The fields in this Model are  
flow, buffer\_plan, produced, dates, quantity, quantity(Date), quantity(Date\_Range), lots, upstream\_flow\_plans, downstream\_flow\_plans, owner.

**Model Item\_Acceptance**  
The fields in this Model are



Summary Section	wait()
-----------------	--------

item\_promise, configuration, item, quantity, delivery\_plan, plan\_to\_satisfy, accept\_as\_promised, plan\_as\_accepted, not\_planned, planned\_late, planned\_early, planned\_short, planned\_excess, last\_change, problems (Date\_Range), problems (Problem\_Category), problems (Date\_Range, Problem\_Category), planned\_date\_problem, planned\_quantity\_problem, owner.

**Model Delivery\_Available\_To\_Promise**  
 The fields in this Model are  
 delivery\_date, item\_atp, offer\_promise, owner.

**Model Item\_Promise**  
 The Fields in this Model are  
 item\_request, item\_acceptance, item\_atp, matching\_forecasts, consumed\_forecast, configuration, item, quantity, list\_price, discount, price, promised\_short, promised\_excess, promised\_overpriced, delivery\_plan, plan\_to\_satisfy, promise\_as\_planned, plan\_as\_promised, receiving\_plan, not\_planned, planned\_late, planned\_early, planned\_short, planned\_excess, last\_change, problems, problems (Date\_Range), problems (Problem\_Category), problems (Date\_Range, Problem\_Category), promised\_quantity\_problem, promised\_price\_problem, planned\_date\_problem, planned\_quantity\_problem, owner.

**Model Item\_Available\_To\_Promise**  
 The Fields in this Model are  
 dates, available, product\_atp, offer\_promise, owner.

**Model Product\_Available\_To\_Promise**  
 The Fields in this Model are  
 product, forecast, forecast\_entry, dates, price, quantity, offered\_quantity, offer\_promise, offer\_promise (Quantity), owner.

**Model Buffer\_Problem\_Detector**  
 The Fields in this Model are  
 detector, feasible, cost, owner.

**Model Lot\_Flow**  
 The Fields in this Model are  
 lot, quantity, consuming\_flow\_plan, upstream\_lot\_flows, downstream\_lot\_flows, owner.

**Model Profile\_Number**

Summary Section	wait()
-----------------	--------

The Fields in this Model are  
 date, value, rate.

**Model Profile\_Percentage**  
 The Fields in this Model are  
 date, value, rate.

**Model Profile\_Quantity**  
 The Fields in this Model are  
 date, value, rate.

**Model Lot**  
 The Fields in this Model are  
 name, remark, quantity (Date), configuration, formed, producing\_flow, consuming\_flows, supplying\_flow, owner.

**Model Ordered\_Sub\_Strategy**  
 The Fields in this Model are  
 sequence, strategy, owner.

**Model Ranked\_Sub\_Strategy**  
 The Fields in this Model are  
 rank, strategy, owner.

**Model Sub\_Item\_Group**  
 The Fields in this Model are  
 item\_group, owner.

**Model Sub\_Item**  
 The Fields in this Model are  
 item, owner.

**Model Load\_Size**  
 The Fields in this Model are  
 name, load\_size\_usage, owner.

**Model Box**  
 The Fields in this Model are  
 specification, x, y, width, height, area, mid\_x, mid\_y, empty, unspecified.

**Model Resource\_Skill**

Summary Section	wait()
The Fields in this Model are skill, efficiency_level (Date), efficiency_level (Date_Range), efficiency, owner.	
Model Skill_Resource The Fields in this Model are resource, efficiency_level (Date), efficiency_level (Date_Range), efficiency_profile (Date_Range), efficiency, owner.	
Model Product_Allocation The Fields in this Model are member, split, owner.	
Model Calendar_Entry The Fields in this Model are name, value, description, effective, daily_start, daily_end, day_pattern, rank, charge, owner.	
Model Calendar The Fields in this Model are name, description, entry_value, entries, entry (Date), dates (Date_Range), sub_calendars.	
Model Sub_Calendar The Fields in this Model are calendar, rank_expression, owner.	
Model Alternate_Operation The Fields in this Model are operation, quantity_per, rank, percentage, owner.	
Model Effective_Calendar_Operation The Fields in this Model are operation, quantity_per, percentage, calendar, owner.	
Model Routing_Operation The Fields in this Model are sequence_number, operation, quantity_per, owner.	
Model Consolidation The Fields in this Model are name, operation_plans, inc (Operation_Plan), dec (Operation_Plan), owner.	

Summary Section	wait()
Model Resource_Setup_Order The Fields in this Model are	
Model Resource_Blocks The Fields in this Model are	
Model Size_Dimension The Fields in this Model are name, min_size, max_size, owner.	
Model Flow_Criterion The Fields in this Model are criterion, rank, owner.	
Model Sorted_Bucket The Fields in this Model are dates, ending_on_hand, lock_ending_on_hand, producing_flow_plans, consuming_flow_plans, owner.	
Model Calendar_Plan The Fields in this Model are calendar, horizon, entry_value, entry (Date), dates (Date_Range).	
Model Worksheet The Fields in this Model are	
Model Control The Fields in this Model are	
Model Connection The Fields in this Model are	
Model User The Fields in this Model are name, full_name, responsibility, remark, organization, members, member_of (User), super_user, data_directories, current_data_directory, load_data_layouts (String), inc_new_data_layout (String, String), report_directories, reports, layouts, worksheets, styles, formats, domains, load_files, connections, active_ui, language,	

Summary Section	wait ( )
-----------------	----------

**Model Report\_Directory**  
 The Fields in this Model are  
 sequence, directory, include, description, editable, owner.

**Model Report**  
 The Fields in this Model are

**Model Layout**  
 The Fields in this Model are

**Model Style**  
 The Fields in this Model are

**Model Format**  
 The Fields in this Model are  
 name\_type, name, directory, description, editable, save, save (Pathname),  
 handles (Type), spec, owner.

**Model Domain**  
 The Fields in this Model are

**Model Model\_Type**  
 The Fields in this Model are  
 name, description, owner\_model, extensible, access, values (Typed\_Value),  
 fields, extension\_selectors.

**Model Field**  
 The Fields in this Model are  
 name, type, description, default, editable, field\_type, access, after\_set,  
 extension\_selector, extensions, user\_defined, value (Typed\_Value), owner.

**Model Extension\_Selector**  
 The Fields in this Model are  
 name, description, extensions, selected\_fields (Symbol), owner.

**Model Field\_Error**  
 The Fields in this Model are  
 target\_model, target\_field, error\_value, error\_input, description, source.

**Model Function**  
 The Fields in this Model are

Summary Section	wait ( )
-----------------	----------

**Model Breakpoint**  
 The Fields in this Model are

## Extensions

**Extension LINK**  
 The Fields in this Extension are  
 managed, locations, top\_locations, items, top\_items, buffers, resources, skills,  
 operations, configurations, item\_groups, top\_item\_groups, buffers\_at\_level (Inte-  
 ger), operations\_at\_level (Integer), buffer\_levels, operation\_levels.

**Extension SUPPLIER**  
 The Fields in this Extension are  
 items.

**Extension CUSTOMER**  
 The Fields in this Extension are

**Extension LINK**  
 The Fields in this Extension are  
 buffer\_plans, resource\_plans, operation\_plans, operation\_states, requests,  
 promises, acceptances, plan\_to\_satisfy\_unanswered\_requests,  
 plan\_to\_satisfy\_queued\_requests, plan\_to\_satisfy\_all\_requests,  
 plan\_to\_satisfy\_all\_promises, promise\_as\_planned, supply\_requests,  
 supply\_promises, supply\_item\_requests, supply\_item\_requests (Date\_Range),  
 supply\_item\_requests (Item), supply\_item\_requests (Item, Date\_Range),  
 item\_demand (List(Item), Date\_Range), problems, problems (Date\_Range), prob-  
 lems (Problem\_Category), problems (Problem\_Category, Date\_Range), prob-  
 lem\_categories.

**Extension SUPPLIER**  
 The Fields in this Extension are  
 requests, promises, acceptances.

**Extension CUSTOMER**  
 The Fields in this Extension are

**Extension ONE**  
 The Fields in this Extension are

Summary Section	wait ( )
-----------------	----------

Extension MONTHS  
The Fields in this Extension are

Extension WEEKS  
The Fields in this Extension are  
first\_day\_of\_week, week\_buckets.

Extension DAYS  
The Fields in this Extension are  
day\_buckets.

Extension DATES  
The Fields in this Extension are  
bucket\_starts.

Extension OPERATION\_PLAN\_RANK\_RANGE  
The Fields in this Extension are  
min\_rank, max\_rank.

Extension OPERATION\_PLAN\_RANK\_EXPRESSION  
The Fields in this Extension are  
expression.

Extension FEASIBILITY  
The Fields in this Extension are  
target\_interaction.

Extension MINIMIZE\_PROBLEM\_COUNT  
The Fields in this Extension are  
target\_problem\_count.

Extension MINIMIZE\_PROBLEMS  
The Fields in this Extension are  
target\_problems.

Extension MINIMIZE\_LATENESS  
The Fields in this Extension are  
target\_lateness.

Extension WEIGHTED\_LATENESS

Summary Section	wait ( )
-----------------	----------

The Fields in this Extension are  
target\_lateness, day\_late\_power, quantity\_late\_power.

Extension MINIMIZE\_SHORTNESS  
The Fields in this Extension are  
target\_shortness.

Extension WEIGHTED\_SHORTNESS  
The Fields in this Extension are  
target\_shortness, quantity\_short\_power.

Extension MINIMIZE\_COST  
The Fields in this Extension are  
target\_cost.

Extension MAXIMIZE\_PROFIT  
The Fields in this Extension are  
target\_profit.

Extension MAXIMIZE\_REVENUE  
The Fields in this Extension are  
target\_revenue.

Extension NO\_PROBLEMS  
The Fields in this Extension are  
problem\_filter.

Extension TARGET\_ACHIEVED  
The Fields in this Extension are

Extension RESOLVE\_COUNT\_EXCEEDED  
The Fields in this Extension are

Extension MANUAL  
The Fields in this Extension are

Extension FEASIBILITY  
The Fields in this Extension are  
total\_interaction.

Extension MINIMIZE\_PROBLEM\_COUNT

Summary Section	Wall ( )
-----------------	----------

The Fields in this Extension are  
**problem\_count.**

Extension **MINIMIZE\_PROBLEMS**  
 The Fields in this Extension are  
**problems.**

Extension **MINIMIZE\_LATENESS**  
 The Fields in this Extension are  
**total\_lateness.**

Extension **WEIGHTED\_LATENESS**  
 The Fields in this Extension are  
**total\_lateness.**

Extension **WEIGHTED\_SHORTNESS**  
 The Fields in this Extension are  
**total\_shortness.**

Extension **MINIMIZE\_SHORTNESS**  
 The Fields in this Extension are  
**total\_shortness.**

Extension **MINIMIZE\_COST**  
 The Fields in this Extension are  
**total\_cost.**

Extension **MAXIMIZE\_PROFIT**  
 The Fields in this Extension are  
**total\_profit.**

Extension **MAXIMIZE\_REVENUE**  
 The Fields in this Extension are  
**total\_revenue.**

Extension **NO\_PROBLEMS**  
 The Fields in this Extension are

Extension **TARGET\_ACHIEVED**  
 The Fields in this Extension are

Summary Section	Wall ( )
-----------------	----------

Extension **RESOLVE\_COUNT\_EXCEEDED**  
 The Fields in this Extension are

Extension **MANUAL**  
 The Fields in this Extension are  
**done.**

Extension **INDIVIDUAL**  
 The Fields in this Extension are  
**product, atp\_entries.**

Extension **GROUP**  
 The Fields in this Extension are  
**product\_group, sub\_forecasts, active\_sub\_forecasts, active\_leaf\_product\_forecasts, use\_std\_split.**

Extension **MOVE\_IN**  
 The Fields in this Extension are

Extension **MOVE\_OUT**  
 The Fields in this Extension are

Extension **SPLIT**  
 The Fields in this Extension are

Extension **USE\_MORE**  
 The Fields in this Extension are

Extension **USE\_LESS**  
 The Fields in this Extension are

Extension **USE\_ALTERNATE\_OPERATION**  
 The Fields in this Extension are  
**min\_alt, max\_alt.**

Extension **USE\_ALTERNATE\_RESOURCE**  
 The Fields in this Extension are

Extension **USE\_EFFECTIVE\_ALTERNATE**  
 The Fields in this Extension are

Summary Section	wait ( )
-----------------	----------

Extension **INCREASE\_CAPACITY**  
The Fields in this Extension are

Extension **DECREASE\_CAPACITY**  
The Fields in this Extension are

Extension **RESIZE\_MORE**  
The Fields in this Extension are

Extension **RESIZE\_LESS**  
The Fields in this Extension are

Extension **UPSTREAM**  
The Fields in this Extension are

Extension **DOWNSTREAM**  
The Fields in this Extension are

Extension **REQUEST\_NOT\_PLANNED**  
The Fields in this Extension are  
item\_request, item\_promise, item\_acceptance, delivery\_request, delivery\_promise, delivery\_acceptance, request, promise, acceptance, delivery\_plan.

Extension **REQUEST\_NOT\_PLANNED**  
The Fields in this Extension are  
item\_request\_filter, problem\_filter.

Extension **REQUEST\_PLANNED\_LATE**  
The Fields in this Extension are  
item\_request, item\_promise, item\_acceptance, delivery\_request, delivery\_promise, delivery\_acceptance, request, promise, acceptance, delivery\_plan.

Extension **REQUEST\_PLANNED\_LATE**  
The Fields in this Extension are  
min\_lateness, item\_request\_filter, problem\_filter.

Extension **REQUEST\_PLANNED\_EARLY**  
The Fields in this Extension are

Summary Section	wait ( )
-----------------	----------

item\_request, item\_promise, item\_acceptance, delivery\_request, delivery\_promise, delivery\_acceptance, request, promise, acceptance, delivery\_plan.

Extension **REQUEST\_PLANNED\_EARLY**  
The Fields in this Extension are  
min\_earliness, item\_request\_filter, problem\_filter.

Extension **REQUEST\_PLANNED\_SHORT**  
The Fields in this Extension are  
item\_request, item\_promise, item\_acceptance, delivery\_request, delivery\_promise, delivery\_acceptance, request, promise, acceptance, delivery\_plan.

Extension **REQUEST\_PLANNED\_SHORT**  
The Fields in this Extension are  
min\_shortness, item\_request\_filter, problem\_filter.

Extension **REQUEST\_PLANNED\_EXCESS**  
The Fields in this Extension are  
item\_request, item\_promise, item\_acceptance, delivery\_request, delivery\_promise, delivery\_acceptance, request, promise, acceptance, delivery\_plan.

Extension **REQUEST\_PLANNED\_EXCESS**  
The Fields in this Extension are  
min\_excess, item\_request\_filter, problem\_filter.

Extension **PROMISE\_NOT\_PLANNED**  
The Fields in this Extension are  
item\_request, item\_promise, item\_acceptance, delivery\_request, delivery\_promise, delivery\_acceptance, request, promise, acceptance, delivery\_plan.

Extension **PROMISE\_NOT\_PLANNED**  
The Fields in this Extension are  
item\_promise\_filter, problem\_filter.

Extension **PROMISE\_PLANNED\_LATE**  
The Fields in this Extension are

Summary Section	wait ()
-----------------	---------

item\_request, item\_promise, item\_acceptance, delivery\_request, delivery\_promise, delivery\_acceptance, request, promise, acceptance, delivery\_plan.

Extension PROMISE\_PLANNED\_LATE  
The Fields in this Extension are  
min\_latency, item\_promise\_filter, problem\_filter.

Extension PROMISE\_PLANNED\_EARLY  
The Fields in this Extension are  
item\_request, item\_promise, item\_acceptance, delivery\_request, delivery\_promise, delivery\_acceptance, request, promise, acceptance, delivery\_plan.

Extension PROMISE\_PLANNED\_EARLY  
The Fields in this Extension are  
min\_earliness, item\_promise\_filter, problem\_filter.

Extension PROMISE\_PLANNED\_SHORT  
The Fields in this Extension are  
item\_request, item\_promise, item\_acceptance, delivery\_request, delivery\_promise, delivery\_acceptance, request, promise, acceptance, delivery\_plan.

Extension PROMISE\_PLANNED\_SHORT  
The Fields in this Extension are  
min\_shortness, item\_promise\_filter, problem\_filter.

Extension PROMISE\_PLANNED\_EXCESS  
The Fields in this Extension are  
item\_request, item\_promise, item\_acceptance, delivery\_request, delivery\_promise, delivery\_acceptance, request, promise, acceptance, delivery\_plan.

Extension PROMISE\_PLANNED\_EXCESS  
The Fields in this Extension are  
min\_excess, item\_promise\_filter, problem\_filter.

Extension ACCEPTANCE\_NOT\_PLANNED  
The Fields in this Extension are

Summary Section	wait ()
-----------------	---------

item\_request, item\_promise, item\_acceptance, delivery\_request, delivery\_promise, delivery\_acceptance, request, promise, acceptance, delivery\_plan.

Extension ACCEPTANCE\_NOT\_PLANNED  
The Fields in this Extension are  
item\_acceptance\_filter, problem\_filter.

Extension ACCEPTANCE\_PLANNED\_LATE  
The Fields in this Extension are  
item\_request, item\_promise, item\_acceptance, delivery\_request, delivery\_promise, delivery\_acceptance, request, promise, acceptance, delivery\_plan.

Extension ACCEPTANCE\_PLANNED\_LATE  
The Fields in this Extension are  
min\_latency, item\_acceptance\_filter, problem\_filter.

Extension ACCEPTANCE\_PLANNED\_EARLY  
The Fields in this Extension are  
item\_request, item\_promise, item\_acceptance, delivery\_request, delivery\_promise, delivery\_acceptance, request, promise, acceptance, delivery\_plan.

Extension ACCEPTANCE\_PLANNED\_EARLY  
The Fields in this Extension are  
min\_earliness, item\_acceptance\_filter, problem\_filter.

Extension ACCEPTANCE\_PLANNED\_SHORT  
The Fields in this Extension are  
item\_request, item\_promise, item\_acceptance, delivery\_request, delivery\_promise, delivery\_acceptance, request, promise, acceptance, delivery\_plan.

Extension ACCEPTANCE\_PLANNED\_SHORT  
The Fields in this Extension are  
min\_shortness, item\_acceptance\_filter, problem\_filter.

Extension ACCEPTANCE\_PLANNED\_EXCESS  
The Fields in this Extension are

Summary Section	wait ()
-----------------	---------

item\_request, item\_promise, item\_acceptance, delivery\_request, delivery\_promise, delivery\_acceptance, request, promise, acceptance, delivery\_plan.

Extension ACCEPTANCE\_PLANNED\_EXCESS

The Fields in this Extension are  
min\_excess, item\_acceptance\_filter, problem\_filter.

Extension SEQUENCE\_RUN\_ONCE

The Fields in this Extension are  
sub\_strategies.

Extension SEQUENCE\_RUN\_MULTIPLE

The Fields in this Extension are  
sub\_strategies.

Extension SEQUENCE\_RUN\_ONCE

The Fields in this Extension are  
sub\_strategies, current\_sub.

Extension SEQUENCE\_RUN\_MULTIPLE

The Fields in this Extension are  
sub\_strategies, current\_sub.

Extension BEFORE\_AND\_AFTER

The Fields in this Extension are  
before\_run, before\_search, after\_resolve, after\_search, after\_success, after\_run.

Extension BEFORE\_AND\_AFTER

The Fields in this Extension are  
before\_run, before\_search, after\_resolve, after\_search, after\_success, after\_run.

Extension SEQUENTIAL\_ALTERNATES

The Fields in this Extension are  
propagation, evaluation, cleanup, min\_alt, max\_alt.

Extension SEQUENTIAL\_ALTERNATES

The Fields in this Extension are  
propagation, evaluation, cleanup, min\_alt, max\_alt, current\_alt.

Extension SEQUENTIAL\_ALTERNATES\_KEEP\_BEST

Summary Section	wait ()
-----------------	---------

The Fields in this Extension are  
propagation, evaluation, min\_alt, max\_alt.

Extension SEQUENTIAL\_ALTERNATES\_KEEP\_BEST

The Fields in this Extension are  
propagation, evaluation, min\_alt, max\_alt.

Extension PROPORTIONAL\_RESOLVES

The Fields in this Extension are  
ranked\_sub\_strategies.

Extension PROPORTIONAL\_RESOLVES

The Fields in this Extension are  
sub\_strategies.

Extension ORDERED\_RESOLVES

The Fields in this Extension are  
sub\_strategies.

Extension ORDERED\_RESOLVES

The Fields in this Extension are  
sub\_strategies.

Extension PRODUCE

The Fields in this Extension are  
buffer\_plan, configuration.

Extension CONSUME

The Fields in this Extension are  
buffer\_plan, configuration.

Extension DELIVER

The Fields in this Extension are  
motive\_request, motive\_promise, planned\_for\_promise, item, configuration.

Extension RECEIVE

The Fields in this Extension are  
item.

Extension PLANNED\_BEFORE\_CURRENT

The Fields in this Extension are



Summary Section	wait ( )
operation_plan.	
Extension <b>PLANNED_BEFORE_CURRENT</b> The Fields in this Extension are operation_plan_filter, problem_filter.	
Extension <b>UNRELEASED</b> The Fields in this Extension are operation_plan.	
Extension <b>UNRELEASED</b> The Fields in this Extension are operation_plan_filter, problem_filter.	
Extension <b>NEEDS_RELEASE</b> The Fields in this Extension are operation_plan.	
Extension <b>NEEDS_RELEASE</b> The Fields in this Extension are operation_plan_filter, problem_filter.	
Extension <b>INCONSISTENT_OPPLAN</b> The Fields in this Extension are operation_plan.	
Extension <b>INCONSISTENT_OPPLAN</b> The Fields in this Extension are operation_plan_filter, problem_filter.	
Extension <b>OPERATION</b> The Fields in this Extension are operation_plan_filter, problem_filter.	
Extension <b>REQUEST_PROMISED_LATE</b> The Fields in this Extension are delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance.	
Extension <b>REQUEST_PROMISED_LATE</b> The Fields in this Extension are	

Summary Section	wait ( )
min_lateness, delivery_promise_filter, problem_filter.	
Extension <b>REQUEST_PROMISED_EARLY</b> The Fields in this Extension are delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance.	
Extension <b>REQUEST_PROMISED_EARLY</b> The Fields in this Extension are min_earliness, delivery_promise_filter, problem_filter.	
Extension <b>REQUEST_PROMISED_SHORT</b> The Fields in this Extension are item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance.	
Extension <b>REQUEST_PROMISED_SHORT</b> The Fields in this Extension are min_shortness, item_promise_filter, problem_filter.	
Extension <b>REQUEST_PROMISED_EXCESS</b> The Fields in this Extension are item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance.	
Extension <b>REQUEST_PROMISED_EXCESS</b> The Fields in this Extension are min_excess, item_promise_filter, problem_filter.	
Extension <b>ITEM_PROMISE_OVERPRICED</b> The Fields in this Extension are item_request, item_promise, item_acceptance, delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance.	
Extension <b>DELIVERY_PROMISE_OVERPRICED</b> The Fields in this Extension are delivery_request, delivery_promise, delivery_acceptance, request, promise, acceptance.	
Extension <b>PROMISE_NOT_OFFERED</b> The Fields in this Extension are	

request, promise, acceptance.

Extension PROMISE\_NOT\_OFFERED  
The Fields in this Extension are  
request\_filter, problem\_filter.

Extension PROMISE\_NOT\_ACCEPTED  
The Fields in this Extension are  
request, promise, acceptance.

Extension PROMISE\_NOT\_ACCEPTED  
The Fields in this Extension are  
request\_filter, problem\_filter.

Extension ACCEPTANCE\_INCONSISTENT  
The Fields in this Extension are  
request, promise, acceptance.

Extension ACCEPTANCE\_INCONSISTENT  
The Fields in this Extension are  
request\_filter, problem\_filter.

Extension REQUEST\_QUEUED  
The Fields in this Extension are  
request, promise, acceptance.

Extension REQUEST\_QUEUED  
The Fields in this Extension are  
request\_filter, problem\_filter.

Extension DELIVERY\_REQUEST\_NOT\_COORDINATED  
The Fields in this Extension are  
delivery\_request, delivery\_promise, delivery\_acceptance, request, promise,  
acceptance.

Extension DELIVERY\_PROMISE\_NOT\_COORDINATED  
The Fields in this Extension are  
delivery\_request, delivery\_promise, delivery\_acceptance, request, promise,  
acceptance.

Extension DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED

The Fields in this Extension are  
delivery\_request, delivery\_promise, delivery\_acceptance, request, promise,  
acceptance.

Extension REQUEST  
The Fields in this Extension are  
request\_filter, problem\_filter.

Extension REQUEST\_PLAN  
The Fields in this Extension are  
item\_request\_filter, problem\_filter.

Extension PROMISE  
The Fields in this Extension are  
promise\_filter, problem\_filter.

Extension PROMISE\_PLAN  
The Fields in this Extension are  
item\_promise\_filter, problem\_filter.

Extension REQUEST\_PROMISE  
The Fields in this Extension are  
delivery\_promise\_filter, problem\_filter.

Extension DELIVERY\_REQUEST\_NOT\_COORDINATED  
The Fields in this Extension are  
problem\_filter.

Extension DELIVERY\_PROMISE\_NOT\_COORDINATED  
The Fields in this Extension are  
problem\_filter.

Extension DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED  
The Fields in this Extension are  
problem\_filter.

Extension NEGATIVE\_ATP  
The Fields in this Extension are  
forecast\_entry.

Extension NEGATIVE\_PLANNED\_ATP

The Fields in this Extension are  
forecast\_entry.

Extension OVER\_COMMITTED  
The Fields in this Extension are  
forecast\_entry.

Extension OVER\_CONSUMED  
The Fields in this Extension are  
forecast\_entry.

Extension UNALLOCATED\_FORECAST  
The Fields in this Extension are  
forecast\_entry.

Extension PROPORTIONAL\_INTERACTION  
The Fields in this Extension are

Extension EARLIEST\_PROBLEM\_START  
The Fields in this Extension are

Extension SORT\_BY\_EXPRESSION  
The Fields in this Extension are

Extension PROPORTIONAL\_INTERACTION  
The Fields in this Extension are

Extension EARLIEST\_PROBLEM\_START  
The Fields in this Extension are

Extension SORT\_BY\_EXPRESSION  
The Fields in this Extension are  
sorting\_criteria.

Extension SUPPLY\_PLANNED\_LATE  
The Fields in this Extension are  
receiving\_plan, item\_request, item\_promise, item\_acceptance, delivery\_request,  
delivery\_promise, delivery\_acceptance, request, promise, acceptance,  
delivery\_plan.

Extension SUPPLY\_PLANNED\_LATE

The Fields in this Extension are  
min\_latency, item\_request\_filter, problem\_filter.

Extension SUPPLY\_PLANNED\_EARLY  
The Fields in this Extension are  
receiving\_plan, item\_request, item\_promise, item\_acceptance, delivery\_request,  
delivery\_promise, delivery\_acceptance, request, promise, acceptance,  
delivery\_plan.

Extension SUPPLY\_PLANNED\_EARLY  
The Fields in this Extension are  
min\_carliness, item\_request\_filter, problem\_filter.

Extension SUPPLY\_PLANNED\_SHORT  
The Fields in this Extension are  
receiving\_plan, item\_request, item\_promise, item\_acceptance, delivery\_request,  
delivery\_promise, delivery\_acceptance, request, promise, acceptance,  
delivery\_plan.

Extension SUPPLY\_PLANNED\_SHORT  
The Fields in this Extension are  
min\_shortness, item\_request\_filter, problem\_filter.

Extension SUPPLY\_PLANNED\_EXCESS  
The Fields in this Extension are  
receiving\_plan, item\_request, item\_promise, item\_acceptance, delivery\_request,  
delivery\_promise, delivery\_acceptance, request, promise, acceptance,  
delivery\_plan.

Extension SUPPLY\_PLANNED\_EXCESS  
The Fields in this Extension are  
min\_excess, item\_request\_filter, problem\_filter.

Extension SUPPLY\_PROMISED\_LATE  
The Fields in this Extension are  
receiving\_plan, item\_request, item\_promise, item\_acceptance, delivery\_request,  
delivery\_promise, delivery\_acceptance, request, promise, acceptance.

Extension SUPPLY\_PROMISED\_LATE  
The Fields in this Extension are  
min\_latency, delivery\_promise\_filter, problem\_filter.

Summary Section	wait()
-----------------	--------

**Extension SUPPLY\_PROMISED\_EARLY**

The fields in this Extension are  
 receiving\_plan, item\_request, item\_promise, item\_acceptance, delivery\_request,  
 delivery\_promise, delivery\_acceptance, request, promise, acceptance.

**Extension SUPPLY\_PROMISED\_EARLY**

The fields in this Extension are  
 min\_curliness, delivery\_promise\_filter, problem\_filter.

**Extension SUPPLY\_PROMISED\_SHORT**

The fields in this Extension are  
 receiving\_plan, item\_request, item\_promise, item\_acceptance, delivery\_request,  
 delivery\_promise, delivery\_acceptance, request, promise, acceptance.

**Extension SUPPLY\_PROMISED\_SHORT**

The fields in this Extension are  
 min\_shortness, item\_promise\_filter, problem\_filter.

**Extension SUPPLY\_PROMISED\_EXCESS**

The fields in this Extension are  
 receiving\_plan, item\_request, item\_promise, item\_acceptance, delivery\_request,  
 delivery\_promise, delivery\_acceptance, request, promise, acceptance.

**Extension SUPPLY\_PROMISED\_EXCESS**

The fields in this Extension are  
 min\_excess, item\_promise\_filter, problem\_filter.

**Extension SUPPLY**

The fields in this Extension are  
 request\_filter, problem\_filter.

**Extension SUPPLY\_PLAN**

The fields in this Extension are  
 item\_request\_filter, problem\_filter.

**Extension SUPPLY\_PROMISE**

The fields in this Extension are  
 delivery\_promise\_filter, problem\_filter.

**Extension FCFS**

Summary Section	wait()
-----------------	--------

The fields in this Extension are

**Extension PER\_ALLOCATED**

The fields in this Extension are  
 retain,

**Extension PER\_COMMITTED**

The fields in this Extension are  
 retain,

**Extension MEMBER\_RANK**

The fields in this Extension are  
 retain, pool\_allocation, reallocate (Plan).

**Extension MEMBER\_RANK**

The fields in this Extension are  
 pooled\_allocated, pooled\_accepted, pooled\_allocated\_available, pooled\_available.

**Extension FIXED\_SPLIT**

The fields in this Extension are  
 retain, allocations.

**Extension SLIDING**

The fields in this Extension are

**Extension HORIZON**

The fields in this Extension are  
 availability\_horizon, buckets (Date\_Range).

**Extension BUCKETED\_ALL**

The fields in this Extension are

**Extension BUCKETED\_ASAP**

The fields in this Extension are

**Extension FIXED**

The fields in this Extension are

**Extension FIXED**

The fields in this Extension are

Summary Section	wait ( )
-----------------	----------

Extension **AT\_END**  
The Fields in this Extension are

Extension **SIMPLE\_FIXED\_QUANTITY**  
The Fields in this Extension are  
order\_quantity, representative\_configuration, representative\_quantity\_per, rough\_fence.

Extension **SIMPLE\_FIXED\_QUANTITY**  
The Fields in this Extension are

Extension **SINGLE\_REQUEST**  
The Fields in this Extension are  
representative\_configuration, representative\_quantity\_per.

Extension **SINGLE\_REQUEST**  
The Fields in this Extension are

Extension **SIMPLE\_FIXED\_TIME**  
The Fields in this Extension are  
interval, representative\_configuration, representative\_quantity\_per, rough\_fence.

Extension **SIMPLE\_FIXED\_TIME**  
The Fields in this Extension are

Extension **WEEKLY**  
The Fields in this Extension are  
day\_of\_week, representative\_configuration, representative\_quantity\_per, rough\_fence.

Extension **WEEKLY**  
The Fields in this Extension are

Extension **DUAL\_REQUEST**  
The Fields in this Extension are  
representative\_configuration, representative\_quantity\_per, rough\_fence.

Extension **DUAL\_REQUEST**  
The Fields in this Extension are

Extension **ON\_TIME**

Summary Section	wait ( )
-----------------	----------

The Fields in this Extension are

Extension **ON\_TIME**  
The Fields in this Extension are

Extension **ON\_TIME**  
The Fields in this Extension are

Extension **FULL\_QUANTITIES\_OF\_ALL\_ITEMS**  
The Fields in this Extension are

Extension **FULL\_QUANTITIES\_OF\_ALL\_ITEMS**  
The Fields in this Extension are

Extension **FULL\_QUANTITIES\_OF\_ALL\_ITEMS**  
The Fields in this Extension are

Extension **UNRESTRICTED**  
The Fields in this Extension are

Extension **UNRESTRICTED**  
The Fields in this Extension are

Extension **UNRESTRICTED**  
The Fields in this Extension are

Extension **NUMBERED**  
The Fields in this Extension are

Extension **NONE**  
The Fields in this Extension are

Extension **NONE**  
The Fields in this Extension are

Extension **FIXED**  
The Fields in this Extension are  
price.

Extension **ON\_TIME**  
The Fields in this Extension are

Summary Section	wait ( )
-----------------	----------

Extension ALL  
The Fields in this Extension are

Extension ALL\_ON\_TIME  
The Fields in this Extension are

Extension ASAP  
The Fields in this Extension are

Extension ASAP\_MONTHLY  
The Fields in this Extension are

Extension BUCKETED\_ALLOCATION  
The Fields in this Extension are

Extension BUCKETED\_ALL\_MIN\_PRICE  
The Fields in this Extension are

Extension BUCKETED\_MIN\_PRICE\_ASAP  
The Fields in this Extension are

Extension SHIP\_IN\_RATIO  
The Fields in this Extension are

Extension UNSPECIFIED  
The Fields in this Extension are

Extension NUMBER  
The Fields in this Extension are  
default\_number.

Extension QUANTITY  
The Fields in this Extension are  
default\_quantity.

Extension NUMBER\_QUANTITY  
The Fields in this Extension are  
default\_number, default\_quantity.

Extension SYMBOL

Summary Section	wait ( )
-----------------	----------

The Fields in this Extension are  
default\_value.

Extension TIME  
The Fields in this Extension are  
default\_time.

Extension ALTERNATES\_PRIMARY  
The Fields in this Extension are  
splittable, alternates.

Extension ALTERNATES\_PRIMARY  
The Fields in this Extension are  
move\_to\_alternate (Operation\_Plan, Operation, Quantity), move\_to\_alternate  
(Operation\_Plan, Operation), move\_to\_alternate (Operation\_Plan),  
move\_to\_alternate.

Extension ALTERNATES\_PROPORTIONAL  
The Fields in this Extension are  
splittable, alternates.

Extension ALTERNATES\_PROPORTIONAL  
The Fields in this Extension are  
move\_to\_alternate (Operation\_Plan, Operation, Quantity), move\_to\_alternate  
(Operation\_Plan, Operation), move\_to\_alternate (Operation\_Plan),  
move\_to\_alternate.

Extension EFFECTIVE\_CALENDAR  
The Fields in this Extension are  
splittable, effective\_alternates.

Extension EFFECTIVE\_CALENDAR  
The Fields in this Extension are  
move\_to\_alternate (Operation\_Plan, Operation, Quantity), move\_to\_alternate  
(Operation\_Plan, Operation), move\_to\_alternate (Operation\_Plan),  
move\_to\_alternate.

Extension CONSUME\_FIXED  
The Fields in this Extension are  
fixed\_quantity.

Summary Section

wait()

Extension CONSUME\_PER

The Fields in this Extension are

quantity\_per.

Extension PRODUCE\_FIXED

The Fields in this Extension are

fixed\_quantity.

Extension PRODUCE\_PER

The Fields in this Extension are

quantity\_per.

Extension PRODUCE\_YIELD

The Fields in this Extension are

quantity\_per, yield, yield\_near, near\_time.

Extension PRODUCE\_YIELD\_CALENDAR

The Fields in this Extension are

quantity\_per, yield, yield\_near, near\_time, calendar.

Extension EARLIEST

The Fields in this Extension are

release\_name, operation, top\_operation.

Extension FIXED

The Fields in this Extension are

fixed\_quantity.

Extension LINEAR

The Fields in this Extension are

linear\_quantity.

Extension RESOURCE

The Fields in this Extension are

Extension ONE

The Fields in this Extension are

Extension UNIDENTIFIED\_OP\_STATE

The Fields in this Extension are

operation\_state.

Summary Section

wait()

Extension UNIDENTIFIED\_OP\_STATE

The Fields in this Extension are

operation\_plan\_filter, problem\_filter.

Extension DELAY\_ONLY\_FIXED

The Fields in this Extension are

time.

Extension DELAY\_ONLY\_FIXED

The Fields in this Extension are

Extension DELAY\_ONLY\_BASIC

The Fields in this Extension are

time, time\_per.

Extension DELAY\_ONLY\_BASIC

The Fields in this Extension are

Extension BASIC\_CALENDARS

The Fields in this Extension are

time\_calendar, time\_per\_calendar.

Extension BASIC\_CALENDARS

The Fields in this Extension are

Extension FIXED\_TIME

The Fields in this Extension are

time.

Extension FIXED\_TIME

The Fields in this Extension are

Extension TIME\_MULTIPLE

The Fields in this Extension are

base\_time, base\_units.

Extension TIME\_MULTIPLE

The Fields in this Extension are

Extension BASIC

The Fields in this Extension are

Summary Section	wait()
-----------------	--------

The fields in this Extension are  
time, time\_per.

Extension BASIC  
The Fields in this Extension are

Extension BASIC\_DELAYED  
The Fields in this Extension are  
time, time\_per, pre\_load\_delay, post\_load\_delay.

Extension BASIC\_DELAYED  
The Fields in this Extension are

Extension REQUEST\_FIXED  
The Fields in this Extension are  
time, order\_lead\_time, item, supplier, item\_name, seller.

Extension REQUEST\_FIXED  
The Fields in this Extension are  
item\_request.

Extension REQUEST\_FIXED\_WITH\_ANALYSIS  
The Fields in this Extension are  
time, order\_lead\_time, item, supplier, item\_name, seller.

Extension REQUEST\_FIXED\_WITH\_ANALYSIS  
The Fields in this Extension are  
item\_request.

Extension ROUTING  
The Fields in this Extension are  
routing\_operations.

Extension ROUTING  
The Fields in this Extension are

Extension STARTED  
The Fields in this Extension are  
item, quantity.

Extension COMPLETED

Summary Section	wait()
-----------------	--------

The Fields in this Extension are  
item, quantity.

Extension IN\_FRONT  
The Fields in this Extension are  
quantity.

Extension SIMPLE\_CONSOLIDATION  
The Fields in this Extension are  
consolidation\_fence.

Extension SIMPLE\_CONSOLIDATION  
The Fields in this Extension are  
consolidations, unconsolidated\_operation\_plans.

Extension UNCONSOLIDATED  
The Fields in this Extension are  
resource\_plan, operation\_plans.

Extension UNCONSOLIDATED  
The Fields in this Extension are  
resource\_plan\_filter.

Extension UNCOORDINATED  
The Fields in this Extension are  
resource\_plan, operation\_plans, consolidation.

Extension UNCOORDINATED  
The Fields in this Extension are  
resource\_plan\_filter.

Extension CONSOLIDATION\_OVERSIZE  
The Fields in this Extension are  
resource\_plan, operation\_plans, consolidation, oversize\_dimensions, oversize (Symbol).

Extension CONSOLIDATION\_OVERSIZE  
The Fields in this Extension are  
resource\_plan\_filter,

Extension CONSOLIDATION\_UNDERSIZE



Summary Section	wait()
-----------------	--------

The Fields in this Extension are resource\_plan, operation\_plans, consolidation, undersize\_dimensions, undersize (Symbol).

Extension CONSOLIDATION\_UNDERSIZE  
The Fields in this Extension are resource\_plan\_filter,

Extension FIXED  
The Fields in this Extension are fixed\_efficiency.

Extension CALENDAR  
The Fields in this Extension are efficiency\_calendar.

Extension INFINITE\_USE  
The Fields in this Extension are

Extension INFINITE\_USE  
The Fields in this Extension are

Extension EXCLUSIVE\_USE  
The Fields in this Extension are

Extension EXCLUSIVE\_USE  
The Fields in this Extension are

Extension SHARED\_USE  
The Fields in this Extension are buckets, bucket\_fence, min\_load, min\_load\_fence, max\_bucket\_load, upstream\_bucket\_pad, downstream\_bucket\_pad.

Extension SHARED\_USE  
The Fields in this Extension are

Extension ZERO  
The Fields in this Extension are

Extension OVERLOAD  
The Fields in this Extension are

Summary Section	wait()
-----------------	--------

resource\_plan, overload\_time, overload\_std\_time.

Extension OVERLOAD  
The Fields in this Extension are resource\_plan\_filter, problem\_filter, min\_overload\_time, min\_overload\_std\_time.

Extension OVERSIZE  
The Fields in this Extension are resource\_plan, oversize.

Extension OVERSIZE  
The Fields in this Extension are resource\_plan\_filter, problem\_filter, min\_oversize.

Extension BUCKET\_OVERSIZE  
The Fields in this Extension are resource\_plan, bucket\_oversize.

Extension BUCKET\_OVERSIZE  
The Fields in this Extension are resource\_plan\_filter, min\_bucket\_oversize.

Extension UNDERLOAD  
The Fields in this Extension are resource\_plan, underload.

Extension UNDERLOAD  
The Fields in this Extension are resource\_plan\_filter, min\_underload.

Extension RESOURCE  
The Fields in this Extension are resource\_plan\_filter, problem\_filter.

Extension FIXED  
The Fields in this Extension are fixed\_efficiency.

Extension CALENDAR  
The Fields in this Extension are efficiency\_calendar.

**Extension PRIMARY**  
The Fields in this Extension are primary.

**Extension PREFER\_PRIMARY**  
The Fields in this Extension are primary.

**Extension EVEN**  
The Fields in this Extension are

**Extension MAX\_EFFICIENCY**  
The Fields in this Extension are

**Extension UNLIMITED**  
The Fields in this Extension are

**Extension FIXED\_COUNT**  
The Fields in this Extension are max\_operation\_count.

**Extension FIXED\_QUANTITY**  
The Fields in this Extension are max\_size.

**Extension CALENDAR\_COUNT**  
The Fields in this Extension are size\_calendar.

**Extension MULTI\_DIMENSION**  
The Fields in this Extension are dimensions.

**Extension FIXED**  
The Fields in this Extension are fixed\_efficiency.

**Extension CALENDAR**  
The Fields in this Extension are efficiency\_calendar.

**Extension ZERO**  
The Fields in this Extension are

**Extension FIXED**  
The Fields in this Extension are upstream\_pad, downstream\_pad.

**Extension NEGATIVE\_ON\_HAND**  
The Fields in this Extension are buffer\_plan, low\_on\_hand, deficit.

**Extension NEGATIVE\_ON\_HAND**  
The Fields in this Extension are buffer\_plan\_filter, problem\_filter, min\_deficit.

**Extension OVER\_FLOW\_LIMIT**  
The Fields in this Extension are buffer\_plan, low\_on\_hand, deficit.

**Extension OVER\_FLOW\_LIMIT**  
The Fields in this Extension are buffer\_plan\_filter, problem\_filter, min\_deficit.

**Extension NEGATIVE\_ON\_HAND\_AT\_END**  
The Fields in this Extension are buffer\_plan, low\_on\_hand, deficit.

**Extension NEGATIVE\_ON\_HAND\_AT\_END**  
The Fields in this Extension are buffer\_plan\_filter, problem\_filter, min\_deficit.

**Extension LOT\_OVER\_CONSUMED**  
The Fields in this Extension are buffer\_plan, shortage\_quantity, shortage\_time.

**Extension LOT\_OVER\_CONSUMED**  
The Fields in this Extension are buffer\_plan\_filter, problem\_filter.

**Extension LOT\_NOT\_CONSUMED**



The Fields in this Extension are  
calendar, quantity\_range, multiple\_quantity, fence, after\_fence\_quantity\_range,  
after\_fence\_multiple\_quantity, default\_min\_on\_hand, min\_on\_hand\_calendar,  
default\_excess\_on\_hand, excess\_on\_hand\_calendar, default\_min\_time,  
min\_time\_calendar, end\_item, rank\_delivery\_operation\_plan\_higher,  
producing\_operation, flow\_plan\_selection\_start, flow\_plan\_selection\_end,  
flow\_plan\_selection\_interval, flow\_plan\_filter, flow\_plan\_rank,  
continue\_flow\_plan\_selection, flow\_plan\_resize\_quantity\_range,  
flow\_plan\_move\_restriction, flow\_plan\_split\_restriction.

Extension PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK

The Fields in this Extension are

max\_target\_profile, max\_target\_profile (Date\_Range), max\_target (Date),  
max\_target (Date\_Range), max\_target (Date\_Range, Logical), safety\_target\_profile,  
safety\_target\_profile (Date\_Range), safety\_target (Date), safety\_target  
(Date\_Range), safety\_target (Date\_Range, Logical), cycle\_on\_hand\_profile,  
cycle\_on\_hand\_profile (Date\_Range), cycle\_on\_hand (Date), cycle\_on\_hand  
(Date\_Range), cycle\_on\_hand (Date\_Range, Logical), excess\_on\_hand\_profile,  
excess\_on\_hand\_profile (Date\_Range), excess\_on\_hand (Date), excess\_on\_hand  
(Date\_Range), excess\_on\_hand (Date\_Range, Logical), low\_on\_hand\_profile,  
low\_on\_hand\_profile (Date\_Range), low\_on\_hand (Date), low\_on\_hand  
(Date\_Range), low\_on\_hand (Date\_Range, Logical).

Extension SUPPLY\_CALENDAR

The Fields in this Extension are  
calendar.

Extension ON\_HAND\_CALENDAR

The Fields in this Extension are  
calendar.

Extension ON\_HAND\_CALENDAR

The Fields in this Extension are  
capacity (Date\_Range).

Extension ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK

The Fields in this Extension are  
calendar, flow\_plan\_selection\_start, flow\_plan\_selection\_end,  
flow\_plan\_selection\_interval, flow\_plan\_filter, flow\_plan\_rank,  
continue\_flow\_plan\_selection, flow\_plan\_resize\_quantity\_range,  
flow\_plan\_move\_restriction, flow\_plan\_split\_restriction.

Extension ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK  
The Fields in this Extension are  
capacity (Date\_Range).

Extension FLOW\_LIMIT\_CALENDAR

The Fields in this Extension are  
calendar.

Extension FLOW\_LIMIT\_CALENDAR

The Fields in this Extension are  
capacity (Date\_Range).

Extension FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK

The Fields in this Extension are

calendar, flow\_plan\_selection\_start, flow\_plan\_selection\_end,  
flow\_plan\_selection\_interval, flow\_plan\_filter, flow\_plan\_rank,  
continue\_flow\_plan\_selection, flow\_plan\_resize\_quantity\_range,  
flow\_plan\_move\_restriction, flow\_plan\_split\_restriction.

Extension FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK

The Fields in this Extension are  
capacity (Date\_Range).

Extension CUSTOMER\_RANK

The Fields in this Extension are  
produce\_separately, default\_rank.

Extension SELLER\_RANK

The Fields in this Extension are  
produce\_separately, default\_rank.

Extension REQUEST\_RANK

The Fields in this Extension are  
produce\_separately, default\_rank.

Extension ACTUAL\_OR\_FORECAST

The Fields in this Extension are  
produce\_separately, default\_rank.

Extension REQUEST\_ISSUED

The Fields in this Extension are

Summary Section	wait ( )
produce_separately, default_rank, default_issued.	
Extension PROMISE_DUE	
The Fields in this Extension are	
produce_separately, default_rank, default_due.	
Extension REQUEST_DUE	
The Fields in this Extension are	
produce_separately, default_rank, default_due.	
Extension DUE_DATE	
The Fields in this Extension are	
produce_separately, default_rank, default_due.	
Extension PROMISED	
The Fields in this Extension are	
produce_separately, default_rank.	
Extension ENTRY_DATE	
The Fields in this Extension are	
produce_separately, default_rank.	
Extension INFINITE	
The Fields in this Extension are	
Extension SUPPLIER	
The Fields in this Extension are	
fence.	
Extension BASIC	
The Fields in this Extension are	
min_time, min_on_hand, excess_on_hand, producing_operation, supplying_operation.	
Extension BASIC	
The Fields in this Extension are	
max_target_profile, max_target_profile (Date_Range), max_target (Date), max_target (Date_Range), max_target (Date_Range, Logical), safety_target_profile, safety_target_profile (Date_Range), safety_target (Date), safety_target (Date_Range), safety_target (Date_Range, Logical), cycle_on_hand_profile, cycle_on_hand_profile (Date_Range), cycle_on_hand (Date), cycle_on_hand	

Summary Section	wait ( )
(Date_Range), cycle_on_hand (Date_Range, Logical), excess_on_hand_profile, excess_on_hand_profile (Date_Range), excess_on_hand (Date), excess_on_hand (Date_Range), excess_on_hand (Date_Range, Logical), low_on_hand_profile, low_on_hand_profile (Date_Range), low_on_hand (Date), low_on_hand (Date_Range), low_on_hand (Date_Range, Logical).	
Extension BASIC_FILTER_AND_RANK	
The Fields in this Extension are	
min_time, min_on_hand, excess_on_hand, producing_operation, supplying_operation, flow_plan_selection_start, flow_plan_selection_end, flow_plan_selection_interval, flow_plan_filter, flow_plan_rank, continue_flow_plan_selection, flow_plan_resize_quantity_range, flow_plan_move_restriction, flow_plan_split_restriction.	
Extension BASIC_FILTER_AND_RANK	
The Fields in this Extension are	
max_target_profile, max_target_profile (Date_Range), max_target (Date), max_target (Date_Range), max_target (Date_Range, Logical), safety_target_profile, safety_target_profile (Date_Range), safety_target (Date), safety_target (Date_Range), safety_target (Date_Range), cycle_on_hand (Date), cycle_on_hand (Date_Range), cycle_on_hand (Date_Range, Logical), excess_on_hand_profile, excess_on_hand_profile (Date_Range), excess_on_hand (Date), excess_on_hand (Date_Range), excess_on_hand (Date_Range, Logical), low_on_hand_profile, low_on_hand_profile (Date_Range), low_on_hand (Date), low_on_hand (Date_Range), low_on_hand (Date_Range, Logical).	
Extension FIXED_QUANTITY	
The Fields in this Extension are	
min_on_hand, min_time, excess_on_hand, quantity, producing_operation, supplying_operation.	
Extension FIXED_QUANTITY	
The Fields in this Extension are	
max_target_profile, max_target_profile (Date_Range), max_target (Date), max_target (Date_Range), max_target (Date_Range, Logical), safety_target_profile, safety_target_profile (Date_Range), safety_target (Date), safety_target (Date_Range), safety_target (Date_Range), cycle_on_hand_profile, cycle_on_hand_profile (Date_Range), cycle_on_hand (Date), cycle_on_hand	

Summary Section	Wall ( )
-----------------	----------

(Date\_Range), cycle\_on\_hand (Date\_Range, Logical), excess\_on\_hand\_profile, excess\_on\_hand\_profile (Date\_Range), excess\_on\_hand (Date), excess\_on\_hand (Date\_Range), excess\_on\_hand (Date\_Range, Logical), low\_on\_hand\_profile, low\_on\_hand\_profile (Date\_Range), low\_on\_hand (Date), low\_on\_hand (Date\_Range), low\_on\_hand (Date\_Range, Logical).

**Extension MULTIPLE**  
The Fields in this Extension are  
min\_on\_hand, excess\_on\_hand, quantity\_range, multiple\_quantity, min\_time, producing\_operation, supplying\_operation.

**Extension MULTIPLE**  
The Fields in this Extension are  
max\_target\_profile, max\_target\_profile (Date\_Range), max\_target (Date), max\_target (Date\_Range), max\_target (Date\_Range, Logical), safety\_target\_profile, safety\_target\_profile (Date\_Range), safety\_target (Date), safety\_target (Date\_Range), safety\_target (Date\_Range, Logical), cycle\_on\_hand\_profile, cycle\_on\_hand\_profile (Date\_Range), cycle\_on\_hand (Date), cycle\_on\_hand (Date\_Range), cycle\_on\_hand (Date\_Range, Logical), excess\_on\_hand\_profile, excess\_on\_hand\_profile (Date\_Range), excess\_on\_hand (Date), excess\_on\_hand (Date\_Range), excess\_on\_hand (Date\_Range, Logical), low\_on\_hand\_profile, low\_on\_hand\_profile (Date\_Range), low\_on\_hand (Date), low\_on\_hand (Date\_Range), low\_on\_hand (Date\_Range, Logical).

**Extension MULTIPLE\_FILTER\_AND\_RANK**  
The Fields in this Extension are  
min\_on\_hand, excess\_on\_hand, quantity\_range, multiple\_quantity, min\_time, producing\_operation, supplying\_operation, flow\_plan\_selection\_start, flow\_plan\_selection\_end, flow\_plan\_selection\_interval, flow\_plan\_filter, flow\_plan\_rank, continue\_flow\_plan\_selection, flow\_plan\_resize\_quantity\_range, flow\_plan\_move\_restriction, flow\_plan\_split\_restriction.

**Extension MULTIPLE\_FILTER\_AND\_RANK**  
The Fields in this Extension are  
max\_target\_profile, max\_target\_profile (Date\_Range), max\_target (Date), max\_target (Date\_Range), max\_target (Date\_Range, Logical), safety\_target\_profile, safety\_target\_profile (Date\_Range), safety\_target (Date), safety\_target (Date\_Range), safety\_target (Date\_Range, Logical), cycle\_on\_hand\_profile, cycle\_on\_hand\_profile (Date\_Range), cycle\_on\_hand (Date), cycle\_on\_hand (Date\_Range), cycle\_on\_hand\_profile (Date\_Range), cycle\_on\_hand (Date\_Range, Logical).

Summary Section	Wall ( )
-----------------	----------

(Date\_Range), cycle\_on\_hand (Date\_Range, Logical), excess\_on\_hand\_profile, excess\_on\_hand\_profile (Date\_Range), excess\_on\_hand (Date), excess\_on\_hand (Date\_Range), excess\_on\_hand (Date\_Range, Logical), low\_on\_hand\_profile, low\_on\_hand\_profile (Date\_Range), low\_on\_hand (Date), low\_on\_hand (Date\_Range), low\_on\_hand (Date\_Range, Logical).

**Extension FIXED\_QUANTITY\_FENCED**  
The Fields in this Extension are  
min\_on\_hand, min\_time, excess\_on\_hand, quantity, fence, after\_fence\_excess\_on\_hand, after\_fence\_quantity, producing\_operation, supplying\_operation.

**Extension FIXED\_QUANTITY\_FENCED**  
The Fields in this Extension are  
max\_target\_profile, max\_target\_profile (Date\_Range), max\_target (Date), max\_target (Date\_Range), max\_target (Date\_Range, Logical), safety\_target\_profile, safety\_target\_profile (Date\_Range), safety\_target (Date), safety\_target (Date\_Range), safety\_target (Date\_Range, Logical), cycle\_on\_hand\_profile, cycle\_on\_hand\_profile (Date\_Range), cycle\_on\_hand (Date), cycle\_on\_hand (Date\_Range), cycle\_on\_hand (Date\_Range, Logical), excess\_on\_hand\_profile, excess\_on\_hand\_profile (Date\_Range), excess\_on\_hand (Date), excess\_on\_hand (Date\_Range), excess\_on\_hand (Date\_Range, Logical), low\_on\_hand\_profile, low\_on\_hand\_profile (Date\_Range), low\_on\_hand (Date), low\_on\_hand (Date\_Range), low\_on\_hand (Date\_Range, Logical).

**Extension FIXED\_TIME**  
The Fields in this Extension are  
produce\_time, min\_on\_hand, min\_time, excess\_on\_hand, producing\_operation, supplying\_operation, supply\_time.

**Extension FIXED\_TIME**  
The Fields in this Extension are  
max\_target\_profile, max\_target\_profile (Date\_Range), max\_target (Date), max\_target (Date\_Range), max\_target (Date\_Range, Logical), safety\_target\_profile, safety\_target\_profile (Date\_Range), safety\_target (Date), safety\_target (Date\_Range), safety\_target (Date\_Range, Logical), cycle\_on\_hand\_profile, cycle\_on\_hand\_profile (Date\_Range), cycle\_on\_hand (Date), cycle\_on\_hand (Date\_Range), cycle\_on\_hand (Date\_Range, Logical), excess\_on\_hand\_profile, excess\_on\_hand\_profile (Date\_Range), excess\_on\_hand (Date), excess\_on\_hand (Date\_Range), excess\_on\_hand (Date\_Range, Logical), low\_on\_hand\_profile, low\_on\_hand\_profile (Date\_Range), low\_on\_hand (Date), low\_on\_hand (Date\_Range), low\_on\_hand (Date\_Range, Logical).

**Extension STANDARD**

The Fields in this Extension are

**Extension STANDARD**

The Fields in this Extension are

**Extension CUSTOM**

The Fields in this Extension are

**Extension CUSTOM**

The Fields in this Extension are

**Extension LFL\_SIMPLE**

The Fields in this Extension are  
producing\_operation, consuming\_operation, supplying\_operation.

**Extension LFL\_BOUNDED**

The Fields in this Extension are  
quantity\_range, rough\_fence, rough\_quantity\_range, producing\_operation,  
consuming\_operation, supplying\_operation.

**Extension MLFL\_BOUNDED**

The Fields in this Extension are  
quantity\_range, rough\_fence, rough\_quantity\_range, producing\_operation,  
consuming\_operation, supplying\_operation.

**Extension MANUAL**

The Fields in this Extension are

**Extension CALENDAR**

The Fields in this Extension are  
units\_of\_safety\_stock, user\_bias, customer\_service\_level, default\_mean\_demand,  
default\_mean\_demand\_time\_bucket, mean\_demand\_calendar,  
default\_standard\_deviation\_demand, standard\_deviation\_demand\_calendar,  
default\_mean\_lead\_time, mean\_lead\_time\_calendar,  
default\_standard\_deviation\_lead\_time, standard\_deviation\_lead\_time\_calendar,  
safety\_factor, safety\_factor (Date), demand\_safety\_stock, demand\_safety\_stock  
(Date), supply\_safety\_stock, supply\_safety\_stock (Date).

**Extension NUMBER**

The Fields in this Extension are  
before\_horizon\_number, after\_horizon\_number, intra\_horizon\_number.

**Extension QUANTITY**

The Fields in this Extension are  
before\_horizon\_quantity, after\_horizon\_quantity, intra\_horizon\_quantity.

**Extension NUMBER\_QUANTITY**

The Fields in this Extension are  
before\_horizon\_number, after\_horizon\_number, intra\_horizon\_number,  
before\_horizon\_quantity, after\_horizon\_quantity, intra\_horizon\_quantity.

**Extension SYMBOL**

The Fields in this Extension are  
before\_horizon\_symbol, after\_horizon\_symbol, intra\_horizon\_symbol.

**Extension TIME**

The Fields in this Extension are  
before\_horizon\_time, after\_horizon\_time, intra\_horizon\_time.

**Extension EVERYDAY**

The Fields in this Extension are

**Extension EVERY\_N\_DAYS**

The Fields in this Extension are  
nth.

**Extension WEEKDAYS**

The Fields in this Extension are

**Extension WEEKENDS**

The Fields in this Extension are

**Extension DAYS\_OF\_WEEK**

The Fields in this Extension are  
days.

**Extension DAY\_OF\_MONTH**

The Fields in this Extension are  
day, months.

Summary Section	wait ( )	Summary Section	wait ( )
Extension DAY_OF_WEEK_OF_MONTH The Fields in this Extension are day, nth, week, months.		The Fields in this Extension are specification.	
Extension DAY_OF_LAST_WEEK_OF_MONTH The Fields in this Extension are day, months.		Extension String The Fields in this Extension are specification.	
Extension DAY_OF_YEAR The Fields in this Extension are day.		Extension Symbol The Fields in this Extension are specification.	
Extension YEARLY The Fields in this Extension are month, day.		Extension Date The Fields in this Extension are specification.	
Extension NUMBER The Fields in this Extension are number.		Extension Date_Range The Fields in this Extension are specification, separator.	
Extension QUANTITY The Fields in this Extension are quantity.		Extension Number The Fields in this Extension are specification.	
Extension NUMBER_QUANTITY The Fields in this Extension are number, quantity.		Extension Percentage The Fields in this Extension are specification.	
Extension SYMBOL The Fields in this Extension are symbol.		Extension Integer The Fields in this Extension are specification.	
Extension TIME The Fields in this Extension are time.		Extension Quantity The Fields in this Extension are specification.	
Extension Void The Fields in this Extension are specification.		Extension Quantity_Range The Fields in this Extension are specification, separator.	
Extension Logical		Extension Time The Fields in this Extension are	



specification.

Extension Restriction

The Fields in this Extension are specification.

Extension Horizon\_Date

The Fields in this Extension are  
time\_format, number\_format, weekday\_format, day\_format,  
rel\_day\_of\_month\_format, day\_of\_month\_format, day\_from\_end\_month\_format,  
DOW\_format, DOW\_of\_week\_format, DOW\_of\_month\_format,  
DOW\_of\_week\_month\_format.

Extension List

The Fields in this Extension are  
delimiter, element\_format, width, truncated\_format.

Extension SIMPLE

The Fields in this Extension are

Extension SELECTOR

The Fields in this Extension are

Extension EXTENDED

The Fields in this Extension are

Extension USER

The Fields in this Extension are  
get\_expression, set\_expression.

5 Models

Supply\_Chain

Site

LINK

SUPPLIER

CUSTOMER

Location

Item

Buffer

Buffer\_Problem\_Detector

Flow\_Criterion

Resource

Resource\_Skill

Size\_Dimension

Resource\_Setup\_Order

Resource\_Blocks

Skill

Skill\_Resource

Operation

Load

Load\_Size

Flow

Operation\_Problem\_Detector

Alternate\_Operation

Effective\_Calendar\_Operation

Routing\_Operation

Configuration

Item\_Group

Sub\_Item\_Group

Sub\_Item

Seller

Site\_Group

Explicit\_Site

Product\_Root

Product\_Supplier

Product\_Item

Models	wait()
--------	--------

Product

Generic\_Product  
Alternate\_Product  
Product\_Allocation  
Product\_Group  
Sub\_Product\_Group  
Sub\_Product

Plan

Site\_Plan  
LINK  
SUPPLIER  
CUSTOMER  
Buffer\_Plan  
Lot  
Sorted\_Bucket  
Resource\_Plan  
Consolidation  
Operation\_Plan  
Load\_Plan  
Flow\_Plan  
Lot\_Flow  
Operation\_State  
Request  
Delivery\_Request  
Item\_Request  
Promise  
Delivery\_Promise  
Delivery\_Available\_To\_Promise  
Item\_Promise  
Item\_Available\_To\_Promise  
Product\_Available\_To\_Promise  
Acceptance  
Delivery\_Acceptance  
Item\_Acceptance  
Seller\_Plan  
Forecast  
INDIVIDUAL

Models	wait()
--------	--------

GROUP

Forecast\_Entry  
ATP\_Entry  
Problem  
Active\_Strategy  
Active\_Problem  
Active\_Goal  
Problem\_Category  
Horizon  
Horizon\_Bucket\_Start  
Strategy  
Problem\_Set  
Strategy\_Change  
Strategy\_Lock  
Strategy\_Goal  
Ordered\_Sub\_Strategy  
Ranked\_Sub\_Strategy

Unit

Unit\_Quantity  
Profile\_Number  
Profile\_Percentage  
Profile\_Quantity  
Box  
Calendar\_Entry  
Calendar  
Sub\_Calendar  
Calendar\_Plan  
Worksheet  
Control  
Connection  
User  
Report\_Directory  
Report  
Layout  
Style  
Format  
Domain

Models	wait ( )
--------	----------

Model\_Type  
Field  
Extension\_Selector  
Field\_Error  
Function  
Breakpoint

Models	Supply_Chain Model
--------	--------------------

### 5.1 Supply\_Chain Model

Supply\_Chain -- *an independent (top-level) model*

#### Supply\_Chain Definition

The Supply\_Chain models a set of Sites (organizational units) that make up a supply chain to be managed and planned. The Sites that make up a Supply\_Chain can be modeled in detail, or may be modeled as a "black box". Detailed site models will model the material flow through the site in great detail by modeling operations, resources, buffers. This will also allow detailed planning of demand placed on them. On the other hand, a "black box" site will only model the requests for manufactured/supplied items or promises to supply those items.

Several Supply\_Chains may be modeled at once. This capability is primarily used for What-If analysis of changes to a Supply\_Chain but it can be also used to model multiple concurrent and independent supply chains within one RHYTHM engine.

#### Example

The existing supply chain network has distribution centers in Boston, Chicago, Dallas and Denver. With the capability of modeling multiple supply chains, you could create a scenario where you can add a distribution center in Los Angeles for the west coast customers and plan the same requests to see what the impact of opening a new distribution center would be on your plan.

#### Plan Definition

Each Supply\_Chain may have one or more Plans. A Plan models what a Supply\_Chain is currently doing, its current condition, and what it plans to do in the future. It holds the "live" data, whereas the Supply\_Chain holds the "master" data. Typically, a Supply\_Chain will have only one Plan. However, What-If analysis on Plans can be very important. In particular, there will typically be one active or "released" Plan and another What-If Plan built from the "released" plan for deciding what to release next.

What-If plans can also model different scenarios of forecasted demand and product mix on the same supply chain.

Each Plan may be simulated. There may be several simulations of each Plan using different stochastic data. (Stochastic simulations are not currently supported.)

The Supply\_Chain model has these submodels :  
Site, Seller, Plan.

The Supply\_Chain model has fields that references these models :  
Site, Seller, Plan.

These models have a field that is a Supply\_Chain model :  
Site, Seller, Plan.

The key field for this model is name  
This model may be extended with user-defined fields.

name - - a Symbol field of model Supply\_Chain  
name Field The name of this Supply\_Chain.  
Default: None -- this is a key field

description - - a String field of model Supply\_Chain  
description Field A description of this Supply\_Chain. For example, it may be a description of what is being explored in this What-If.  
Default: none

sites - - a list of Site submodels of model Supply\_Chain  
sites Field The Sites (organizational units) that make up this Supply\_Chain. All demand (Requests and Promises) is placed between Sites. The activity within a Site may be modeled or not. When modeled, it may be modeled at various levels of detail.

top\_sites - - a List(Site) field of model Supply\_Chain  
top\_sites Field A subset of 'sites' that only includes Sites whose 'organization' is [unspecified]. These Sites are the topmost organizations; they are not within another organization. This list is particularly useful as a starting point for displaying the hierarchy of Sites.  
Properties: Export-Only Field

sellers - - a list of Seller submodels of model Supply\_Chain  
sellers Field The sales personnel and sales organizations responsible for forecasting and selling the Products of this Supply\_Chain. A single sales organization may sell Products for one or many of the Sites in the Supply\_Chain. Each Seller consists of a forecast for each of the Products it sells. Based upon those forecasts, demand is placed upon the 'sites', production and distribution 'plans' are created, and the resultant supply is allocated to the Sellers for promising to the customers as the actual demand arrives.

top\_sellers - - a List(Seller) field of model Supply\_Chain  
top\_sellers Field A subset of 'sellers' that only includes Sellers whose 'organization' is [unspecified]. These Sellers are the topmost organizations; they are not within another organization. This list is particularly useful as a starting point for displaying the hierarchy of Sellers.  
Properties: Export-Only Field

plans - - a list of Plan submodels of model Supply\_Chain  
plans Field The various Plans that have been built for managing the 'owner' Supply\_Chain. Each Plan contains a Site\_Plan for each Site and a Seller\_Plan for each Seller in this Supply\_Chain. The various 'plans' can represent alternative scenarios and alternative ways to respond to those scenarios. In that way, several "What If..." Plans can be created, manipulated, and compared intelligently.

Models	Site Model
--------	------------

### 5.1.1 Site Model

Site -- a submodel of model Supply\_Chain

The Site models an organizational unit to be planned. Each Site has independent namespace, authority, privacy, etcetera. Note that a Site could correspond to a plant, to a portion of a plant, to several plants, or to a plant, warehouses, distribution centers, and stores. The Site is not a physical division, but rather an organizational one. It defines a portion of the Supply\_Chain that is planned and controlled by one team of decision makers.

Note however that a team of decision makers is not limited to one Site. Several Sites may all be designated as 'managed' by a single RHYTHM SCP engine. Thus, if a single team of planners has authority over two plants that need separate namespaces (e.g., they use the same Resource names to mean different things), then an independent Site can be setup for each plant, but both Sites will have 'managed' set "true".

Demand between 'managed' and non-'managed' Sites is placed formally as Requests, for which they receive Promises back. The Requests and Promises embody agreements between those separately managed organizations. The promising Site makes Plans to fulfill those Promises. The requesting Site makes Plans assuming those Promises will be fulfilled.

In contrast, demand within a Site and demand between 'managed' Sites is propagated directly and transparently, since it is all under the control of one team of decision makers. All of the elements of the 'managed' Sites are planned as an integrated whole, as a single unit.

Fundamentally, a Site consists of Operations, Resources, and Buffers, which together form the FLO (Flow-Load-Operation) network. Each Sites' FLO network is independent (no shared Resources or Buffers).

The FLO network representation is designed to integrate material and capacity planning, to integrate master and execution planning, and to integrate factory and distribution planning. It is the foundation for achieving Truly Integrated Planning. Furthermore, the FLO network was designed to provide adequate extensibility such that it can be optimally suited for each part of the supply chain. Most supply chains consist of Sites with very different needs (a steel factory feeds a repetitive bearings maker that feeds a specialized products maker which feeds a consumer products maker which feeds a warehouse which feeds a distribution center which feeds a retailer). A software system designed to integrate the supply chain planning should be the best tool for each Site, as well as the best at the inter-site issues.

Models	Site Model
--------	------------

Note that if you are looking for the traditional routing model, look in Operation. The Operation model is flexible and powerful enough to model routings as well as steps in routings. Similarly, the Operation model covers transform and movement of Items, and thus covers Bills-of-Materials and Bills-of-Distribution -- see Operation and, in particular, the Flow submodel within Operation.

If you are looking for a representation of SKUs (stockkeeping units), look at Buffer (which models the flow of an Item at a Location). In fact, many of the characteristics found in traditional systems' Item Master files are moved from the Item model to the Buffer model, allowing variation at different locations (critical for integrated supply chain support).

The "live" and "planning" data is held in the corresponding models suffixed with "\_Plan". For example, the Resource\_Plan models the planned load, maintenance, setup, etc., of a Resource. The Buffer\_Plan models the planned flow into and out of a Buffer. The Operation\_Plan models the state and plans for Operations to be performed. See the Site\_Plan model for more detail on the "\_Plan" models corresponding to the Site's models.

Note that certain naming conventions have been strictly followed to increase consistency and reduce ambiguity. For example, within the FLO Network within a Site, the terms "produce" and "consume" are used. An Operation consumes from some Buffers and produces into other Buffers. Of course, an Operation may produce into a Buffer without building anything, it may just move the Items, or even just re-grade them (use a 150MHz chip as a 100MHz one). Outside of the Site, for the inter-site relationships, the terms "supply" and "purchase" (or "supplier" and "customer") are used. Thus, you will have "supplying Sites" and "producing Operations", but never the other way around.

The model has selectors:

role,

The Site model has fields that references these models :

Site, Site\_Plan, Supply\_Chain.

These models have a field that is a Site model :  
Supply\_Chain, Site, Operation, Resource, Buffer, Product\_Root, Product, Site\_Plan, Location, Item, Skill, Configuration, Item\_Group, Explicit\_Site, Product\_Supplier, REQUEST\_FIXED, REQUEST\_FIXED\_WITH\_ANALYSIS, REQUEST\_FIXED\_ALTERNATES.

Models

Site Model

The key field for this model is name  
This model may be extended with user-defined fields.

**name** - - *a Symbol field of model Site*

The name of this Site.

Default: None -- this is a key field

**description** - - *a String field of model Site*

A description of this Site.

Default: none

**category** - - *a Symbol field of model Site*

A simple user-defined categorization of Sites. This is often used in reports. This will allow users to filter the Sites by category, and to write expressions that depend upon the category of the Site.

The standard reports look for the values "Manufacturing" and "Distribution".

Default: Manufacturing

**sub\_category** - - *a Symbol field of model Site*

A simple user-defined categorization of Sites. This is often used in reports. This will allow users to filter the Sites by category, and to write expressions that depend upon the category of the Sites.

The standard reports don not look for any specific values. It is commonly used to categorize by level or function within the Supply Chain. For example, in Distribution, this may be "Retail", "Customer DC", "Regional DC", "Warehouse", or "Packaging". But any values can be used -- whatever is useful for categorization in reports.

Default: [unspecified]

**rank** - - *a Number field of model Site*

Rank of this site. This is used to compute the planning priority of an item\_rap in active\_strategy.

Default: 0

**organization** - - *a Site field of model Site*

The organization Site of which this Site is a member. 'organization' and 'members', these two fields together provides ability to model hierarchical site organization.

Default: [unspecified]

Models

Site Model

**members** - - *a List(Site) field of model Site*

The Sites which specify this Site as their 'organization'.

Properties: Export-Only Field

**member\_of** (Explicit Site) - - *a Logical field of model Site*

Returns "true" if this Site or any of its 'organization's is the parameter Site. More precisely, if this Site is the parameter Seller, then it returns "true"; otherwise, if this Site's 'organization' is [unspecified], then it returns "false"; otherwise, it returns 'organization.member\_of(parameter)'.  
Properties: Export-Only Field

**role** - - *an extension selector of model Site*

Defines the role this Site plays in the Supply\_Chain: SUPPLIER, LINK, or CUSTOMER. The LINK Sites of the Supply\_Chain transform items obtained from SUPPLIER Sites into the items requested by CUSTOMER Sites. The SUPPLIER Sites introduce material into the Supply\_Chain. The CUSTOMER Sites remove material from the Supply\_Chain. The LINK Sites transform and move material within the Supply\_Chain.

LINK Sites can be modeled in detail. They consist of Operations which use Resources to transform or transfer items between Buffers (the FLO network). They place Requests on SUPPLIER and other LINK Sites, and provide Promises to CUSTOMER and other LINK Sites.

CUSTOMER Sites are simply the delivery destinations of LINK Sites. CUSTOMER

Sites place Requests upon LINK Sites and receive back Promises.

SUPPLIER Sites are similarly the sources of Promises for the Requests from the

LINK Sites.

Default: CUSTOMER

Extensions:

LINK, SUPPLIER, CUSTOMER.

**margin\_target** (Date, Date) - - *a Money field of model Site*

The desired difference between the price received for Products and the marginal cost of producing those items during the given Date\_Range (from start Date to end Date). The purpose of this field is to account for the fixed costs involved with having, maintaining, and running this Site. A profit that covers the marginal costs is not sufficient. Thus, the difference between the price and the marginal costs (the margin) must be large enough to cover the fixed costs and result in a strategically desirable profit. This is computed in strategic planning and is an input to master and operational planning.

THIS IS JUST A SKETCH, as are all cost and margin fields currently.  
Properties:    Export-Only Field

delivery\_name - - a String field of model Site

The name of the customer for delivery.

Default: none

delivery\_contact - - a String field of model Site

The name of the individual to contact concerning deliveries.

Default: none

delivery\_phone - - a String field of model Site

The phone number for delivery.

Default: none

delivery\_fax - - a String field of model Site

The fax phone number for delivery.

Default: none

delivery\_address - - a String field of model Site

The street address for delivery.

Default: none

delivery\_city - - a String field of model Site

The city for delivery.

Default: none

delivery\_state - - a String field of model Site

The state or province for delivery.

Default: none

delivery\_country - - a String field of model Site

The country for delivery.

Default: none

delivery\_postal\_code - - a String field of model Site

The postal code for delivery.

Default: none

billing\_name - - a String field of model Site

The contact name for billing.

Default: none

billing\_contact - - a String field of model Site

The name of the individual to contact for billing.

Default: none

billing\_phone - - a String field of model Site

The phone number for billing.

Default: none

billing\_fax - - a String field of model Site

The fax phone number for billing.

Default: none

billing\_address - - a String field of model Site

The street address for billing.

Default: none

billing\_city - - a String field of model Site

The city for billing.

Default: none

billing\_state - - a String field of model Site

The state for billing.

Default: none

billing\_country - - a String field of model Site

The country for billing.

Default: none

billing\_postal\_code - - a String field of model Site

The postal code for billing.

Default: none

tax\_identification - - a String field of model Site

The Customer's tax identification number for reduced tax.

Default: none

site\_plan (Plan) - - a Site\_Plan field of model Site

The Site\_Plan for this Site in the specified Plan.

Properties:    Export-Only Field

Models	Site Model
--------	------------

owner -- a Supply\_Chain field of model Site

Properties:    Export-Only Field

Models	Standard Extensions of model Site
--------	-----------------------------------

5.1.1.1    Standard Extensions of model Site

5.1.1.1.1    role extensions of model Site

5.1.1.1.1    LINK -- a role extension of model Site

A LINK Site is a "link" in this supply "chain". It purchases Items from zero or more other Sites, performs Operations on Items, moving or transforming them, and supplies Items to zero or more other Sites.

LINK Sites can be modeled in detail. They consist of Operations which use Resources to transform or transfer Items between Buffers (the FLO network). They place Requests on other LINK and SUPPLIER Sites, and provide Promises to other LINK and CUSTOMER Sites.

The LINK model has these submodels :

Location, Item, Buffer, Resource, Skill, Operation, Configuration, Item\_Group.

The LINK model has fields that references these models :

Location, Item, Buffer, Resource, Skill, Operation, Configuration, Item\_Group.

managed -- a Logical field of model LINK

If "true", then this LINK Site is planned and managed by this model. The plans created for such a Site are assumed to be the plans it will follow.

If "false", then this LINK Site is managed elsewhere and this model is simply modeling its behavior. Such modeling is purely informational -- the plans created for such a Site cannot be assumed to be reliable. Only the Promises provided from the managers of that Site can be relied upon (hopefully, anyway).

Requests and Promises will be created between all unmanaged Sites and the managed Sites, just as would be created with SUPPLIER Sites and CUSTOMER Sites. In contrast, managed Sites will interface directly to each other. There is no sense in a planner making Requests to himself or responding with formal Promises to himself. Requests and Promises will be created for informational and reporting purposes, but they will be maintained automatically -- as if there was no Site boundary at all.

Default:    true

locations -- a list of Location submodels of model LINK

The Locations within this Site.



Models	LINK Extension
--------	----------------

**top\_locations** - - *a List(Location) field of model LINK*  
 The Locations within this Site whose 'super\_location' is [unspecified]. These are the topmost Locations at this Site. This is a useful starting point for generating a Location hierarchy.  
 Properties:   Export-Only Field

**items** - - *a list of Item submodels of model LINK*  
 The Items consumed, used, and/or produced at this Site. Requests placed on this Site must be for Items in this list that are 'sellable'.  
 Properties:   Export-Only Field

**top\_items** - - *a List(Item) field of model LINK*  
 The Items within this Site whose 'family' is [unspecified]. These are the topmost Locations at this Site. This is a useful starting point for generating an Item hierarchy.  
 Properties:   Export-Only Field

**buffers** - - *a list of Buffer submodels of model LINK*  
 The Buffers to be used for managing the flow of Items at this Site.  
 Properties:   Export-Only Field

**resources** - - *a list of Resource submodels of model LINK*  
 The Resources which model the capacity to perform Operations at this Site.  
 Properties:   Export-Only Field

**skills** - - *a list of Skill submodels of model LINK*  
 The Skills needed to perform Operations. Resources have these Skills.  
 Properties:   Export-Only Field

**operations** - - *a list of Operation submodels of model LINK*  
 The Operations that transform Items into other Items and/or move Items between Buffers.  
 Properties:   Export-Only Field

**configurations** - - *a list of Configuration submodels of model LINK*  
 Should be hidden. The list of these is not relevant.  
 Properties:   Export-Only Field

**item\_groups** - - *a list of Item\_Group submodels of model LINK*  
 Item\_Groups group items into hierarchies. Each item can appear in at most one Item\_Group in a hierarchy of Item\_Groups. But each item can appear in any number of independent Item\_Group hierarchies.  
 Properties:   Export-Only Field

**top\_item\_groups** - - *a List(Item\_Group) field of model LINK*  
 This List is a subset of item\_groups consisting only of Item\_Groups that are the top of a Item\_Group hierarchy, the Item\_Groups that do not appear within any other Item\_Group.  
 Properties:   Export-Only Field

Models	SUPPLIER Extension
--------	--------------------

**buffers\_at\_level (Integer)** - - *a List(Buffer) field of model LINK*  
 A List of the Buffers at a particular Integer 'level'. Note, site.buffers\_at\_level(3) is equivalent to site.buffers.filter('#level == 3)', but is likely less expensive to compute.  
 Properties:   Export-Only Field

**operations\_at\_level (Integer)** - - *a List(Operation) field of model LINK*  
 A List of the Operations at a particular Integer 'level'. An operation level is defined as the min buffer level of all its consuming flow buffers.  
 Properties:   Export-Only Field

**buffer\_levels** - - *a List(Integer) field of model LINK*  
 A List of all the level Integer values used by any of buffers'. It is a List containing 0, and the each Integer in order up to the largest 'level' code of any of buffers'. This List is generally used to feed the 'buffers\_at\_level' function in order to generate a table of the Buffers by-level.  
 Properties:   Export-Only Field

**operation\_levels** - - *a List(Integer) field of model LINK*  
 A List of all the level Integer values used by any of 'operations'. This List is generally used to feed the 'operations\_at\_level' function in order to generate a table of the Operations by-level.  
 Properties:   Export-Only Field

### 5.1.1.1.2 SUPPLIER -- a role extension of model Site

A SUPPLIER Site is not planned or modeled in detail; rather it represents the Items that can be procured from a supplier by LINK Sites, the Requests for those Items, and the Promises received from the supplier.  
 Properties:   Export-Only Field

The SUPPLIER model has these submodels :  
 Item.

The SUPPLIER model has fields that references these models :  
 Item.

**items** - - *a list of Item submodels of model SUPPLIER*  
 The Items produced at this Site. Requests placed on this Site must be for one of these Items.  
 Properties:   Export-Only Field

5.1.1.1.3  
CUSTOMER -- a role extension of model Site

A CUSTOMER Site is not planned or modeled in detail; rather, it is simply a destination for deliveries and source of Requests.

5.1.1.2 role submodels of model Site  
5.1.1.3 Location Model

Location -- a submodel of model Site

A physical location within a surrounding "super" Location. The 'sub\_locations' are considered 'within' their 'super\_location'. Thus, the Locations form a hierarchy which can be arbitrarily deep.

The Location's coordinates are relative to the others within the same 'super\_location'. The 'super\_location' and all its 'sub\_locations' are placed as a group according to the 'super\_location's coordinates. Note that the coordinates are purely informational, to support graphical Map's of the Locations. The coordinates have no effect on planning. See TRANSIT Operations for computation of transit Time.

In each Site there is one special undeletable Location named [unspecified], which indicates no Location has been specified. This is typically the default value for Location fields. It is treated as if it is everywhere, anywhere, or nowhere, as is appropriate.

The Location model has fields that references these models :  
Location, Box, Site.

These models have a field that is a Location model :  
Resource, Resource\_Plan, Buffer, LINK, Location, Load, TRANSIT.

The key field for this model is name  
This model may be extended with user-defined fields.

name -- a Symbol field of model Location  
The name of this Location. This 'name' must be unique among Locations at this Site or empty (unnamed).

Default: None -- this is a key field

description -- a String field of model Location  
A description of this Location.

Default: none

category -- a Symbol field of model Location  
A simple user-defined categorization of Locations. This is often used in reports. This will allow users to filter the Locations by category, and to write expressions that depend upon the category of the Location.

Models	Location Model
--------	----------------

The standard reports look for the values "Manufacturing", "Distribution", and "Inventory".

Default: Manufacturing

sub\_category - - *a Symbol field of model Location*

A simple user-defined categorization of Locations. This is often used in reports. This will allow users to filter the Locations by category, and to write expressions that depend upon the category of the Locations.

The standard reports don not look for any specific values. It is commonly used to categorize by mode, method, duration, cost, or usage pattern. For example, in Distribution, this may be "Dock", "Receiving", "Staging", "Shelf", or "Lane". But any values can be used -- whatever is useful for categorization in reports.

Default: [unspecified]

super\_location - - *a Location field of model Location*

The Location inside of which this one is located. All the coordinates of this Location are relative to the inside of its super\_location.

Default: [unspecified]

sub\_locations - - *a List[Location] field of model Location*

The Locations that are immediately within this Location. The Locations that specify this Location as their super\_location.

Properties: Export-Only Field

within (Location) - - *a Logical field of model Location*

Returns "true" if this Location is within the argument Location. It will return "true" if passed its super\_location, or its super\_location's super\_location, and so on up to and including [unspecified].

For example, if dock is Location "Loading Dock 1-3" which is in Building One, and building is Location "Building One", then dock.within(building) returns "true", whereas building.within(dock) returns "false". Note that all Locations are within "[unspecified]", and "[unspecified]" is within all other Locations.

Properties: Export-Only Field

box - - *a Box field of model Location*

box of this Location within the super\_location. Note that this is purely informational, to support graphical Maps of the Locations. Its value has no effect on planning. See TRANSIT Operations for computation of transit Time.

Default: unspecified

Models	Location Model
--------	----------------

Properties: command=True

x\_position - - *a Quantity field of model Location*

X-Coordinate of this Location within the super\_location. Note that this is purely informational, to support graphical Maps of the Locations. Its value has no effect on planning. See TRANSIT Operations for computation of transit Time.

Default: 0

Properties: command=True

x\_size\_min - - *a Quantity field of model Location*

The minimum size of this Location along the X-axis. If Locations within this one have larger x\_position's, then x\_size will be larger. That is, every Location is large enough to hold all the Location positions within it. This allows a minimum size to be specified when the Location is larger than indicated by its children (it may not even have children).

Default: 0

Properties: command=True

x\_size - - *a Quantity field of model Location*

The size of this Location along the X-axis. Note that this is purely informational, to support graphical Maps of the Locations. Its value has no effect on planning. See TRANSIT Operations for computation of transit Time.

Properties: command=True Export-Only Field

y\_position - - *a Quantity field of model Location*

Y-Coordinate of this Location within the super\_location. Note that this is purely informational, to support graphical Maps of the Locations. Its value has no effect on planning. See TRANSIT Operations for computation of transit Time.

Default: 0

Properties: command=True

y\_size\_min - - *a Quantity field of model Location*

The minimum size of this Location along the Y-axis. If Locations within this one have larger y\_position's, then y\_size will be larger. That is, every Location is large enough to hold all the Location positions within it. This allows a minimum size to be specified when the Location is larger than indicated by its children (it may not even have children).

Default: 0

Properties: command=True

Models	Item Model
--------	------------

*y\_size* - - *a Quantity field of model Location*  
The size of this Location along the Y-axis. Note that this is purely informational, to support graphical Maps of the Locations. Its value has no effect on planning. See TRANSIT Operations for computation of transit Time.  
Properties:    *command=True*    *Export-Only Field*

*owner* - - *a Site field of model Location*

Properties:    *Export-Only Field*  
**5.1.1.4 Item Model**

*Item* -- *a submodel of model Site*

An Item models a kind of material, part, component, subassembly, assembly, or good which has particular characteristics that define how it can be built, stored, processed, or used.

Note that some of the traditional fields of Item have been split out into the Buffer model. This is important for supply chain modeling and for integrated material and capacity planning. The management of an Item at a particular Location (a SKU) is modeled by Buffer. Any Location-specific or SKU-specific information of an Item is modeled by Buffer.

Further, any pricing, availability, or forecasting information is split out into the Product model. This is done because modern pricing and marketing programs involve much more than just the physical Item -- the Customer and the Order Lead Time may also be at issue.

In each Site, there is a special uneditable Item named [unspecified]. It has the default values for all fields.

The model has selectors:  
*spec.*

The Item model has fields that references these models :  
*Item, Operation, Unit, Site.*

These models have a field that is a Item model :

Models	Item Model
--------	------------

Buffer, LINK, Item, Configuration, SUPPLIER, Product\_Item, Item\_Request, Item\_Acceptance, Item\_Promise, Sub\_Item, DELIVER, RECEIVE, STORAGE, REQUEST\_FIXED, REQUEST\_FIXED\_WITH\_ANALYSIS, REQUEST\_FIXED\_ALTERNATES, STARTED, COMPLETED, Feature.

The key field for this model is *name*  
This model may be extended with user-defined fields.

*name* - - *a Symbol field of model Item*  
The name of this Item. This 'name' must be unique among Items at this Site.  
Default:    *None* -- this is a key field

*description* - - *a String field of model Item*  
A description of this Item.  
Default:    *none*

*category* - - *a Symbol field of model Item*  
A simple user-defined categorization of Items. This is often used in reports. This will allow users to filter the Items by category, and to write expressions that depend upon the category of the Items.

The standard reports look for the values "Purchased", "Intermediate", "End", "Phantom", "Resource", and "Cash". The first four are the standard APICS definitions (though there are common synonyms: "Raw Material" for "Purchased" Items, "WIP" for "Intermediate" Items, and "Finished Goods" for "End" Items). "Resource" Items are typically either renewable Resources (e.g., drill bits), Resources that travel with the material (e.g., molds, dies, mandrill), or Resources that are flow-limited rather than load-limited (e.g., subcontractor that commits to a certain quantity). A "Cash" Item can be used to model cash flow.

Default:    *End*

*drawing\_id* - - *a Symbol field of model Item*  
An identifier for a drawing or graphic of this Item.  
Default:    *none*

*family* - - *a Item field of model Item*  
The Item family to which this Item belongs. An Item family is generally an 'artificial' Item that is used for categorization and grouping. It is useful in generating Reports that include numerous related Items. It does not otherwise effect planning computations. If [unspecified], then this Item is a 'top\_item' of the 'owner' Site.  
Default:    [unspecified]

Models	Item Model
--------	------------

children -- a List(Item) field of model Item  
This List contains each Item that specifies this Item as its 'family'. If this list is empty then it is not an Item family.  
Properties: Export-Only Field

artificial -- a Logical field of model Item  
If "true", then this Item is not real -- it cannot be built, purchased, or obtained -- it does not really exist. For example, an Item family often an artificial Item, it just represents some common characteristics of a family of Items.  
Default: true

spec -- an extension selector of model Item  
The 'spec' extension defines the additional specification fields required in a Configuration of this Item. It defines interchangeability of these Items.

For instance, an **OPTIONED** Item has options that must be specified for different features. The chosen options must match for two Lots of an **OPTIONED** Item to be interchangeable. A **STANDARD** Item has no Configuration fields to be specified -- they are all considered identical (interchangeable). In contrast, a **CUSTOM** Item also has no Configuration-specific information, but they are all considered distinct (not interchangeable).

Note that currently, **RHYTHM** only implements Items with **STANDARD** configuration. Future releases will support more complex configuration related extensions.

Without this extra information, you may not know what you really have. The fields of this Item model provide the default values to be used by the Configuration.

Default: **STANDARD**  
Extensions:  
**STANDARD, CUSTOM,**

lots\_tracked -- a Logical field of model Item  
If "false", then the identity of Lots are not tracked. Once the lot reaches a Buffer, its quantity is lumped in with the rest.

If "true", then the identity of Lots are tracked. Buffers containing this Item will maintain a list of the Lots contained within. The purpose of this is to maintain lot-specific information, such as the Configuration, an expiration Date, concentrations, quality grades, test results, or source information. Such information may be used by downstream Operations in computing behavior.

Models	Item Model
--------	------------

Note that if the 'spec' extension requires additional fields, then 'lots\_tracked' will be "true" (and unsettable to "false"). However, if the 'spec' does not require added fields, this may still be set to "true" in order to preserve user-defined or other lot-specific fields.  
Default: true if 'spec' requires additional fields, false otherwise

delivery -- a Operation field of model Item  
The Operation or service performed to deliver this Item to the customer. This Operation must handle a **DELIVERY** motive. During planning, when operation plans are created for a specific operation, motive specifies reasons for creating those operation plans. For example, a transportation operation may be given a **TRANSIT** motive to transport an item I1 from location A to location B. If the transportation operation can't handle **TRANSIT** motive, no operation plans can be planned to transport item I1 using that operation. Similarly, delivery operation of this item must handle **DELIVERY** motive else, during planning no operation plans will be planned to deliver this item.

This Operation's primary role is to move the Item from a finished goods inventory Buffer to the delivery\_address for that Delivery\_Request.

A secondary role of this Operation is to choose which finished goods inventory Buffer to draw from if there is more than one that supplies this Item. Thus, this may be an **ALTERNATES\_\*** Operation that contains a sub\_operation for each alternate Buffer. The **ALTERNATES\_\*** Operation may choose based upon the delivery\_address of the Delivery\_Request. For example, a Delivery\_Request destined for Los Angeles may be supplied first from San Francisco, alternately from Dallas, and as a last resort from Atlanta.  
Default: [unspecified]

buffers -- a List(Buffer) field of model Item  
All Buffers of the 'owner' Site that manage this Item.  
Properties: Export-Only Field

buffer\_plans (Plan) -- a List(Buffer\_Plan) field of model Item  
All Buffer\_Plans in the specified Plan for the Buffers of the 'owner' Site that manage this Item. This 'buffer\_plans(plan)' is a shortcut for 'buffers\_for\_each(#buffer\_plan(plan))'.  
Properties: Export-Only Field

unit -- *a Unit field of model Item*  
The unit defines the Quantity of one unit of this Item, whether this Item is 'discrete' (can only exist in whole units), and what the preferred\_measure for this Item is. It can also convert between different units of Measure and units of this Item.

For example, one unit of Item X may be "3 kg", "400 ml", and "\$12". It may be 'discrete', meaning that a need for "5 kg" will result in "6 kg" (2 units) being built. The preferred\_measure may be "kg" which will cause most fields that display a Quantity of this Item to display in "kg" by default.

Note that this field is not settable -- you cannot assign it a new Unit. Rather, you can get the Unit of this Item and set that Unit's fields.

Properties: Export-Only Field

owner -- *a Site field of model Item*  
The Site to which this Item belongs.  
Properties: Export-Only Field

5.1.1.4.1 spec submodels of model Item  
5.1.1.5 Buffer Model

Buffer -- *a submodel of model Site*

A Buffer models the management of the flow of interchangeable Items. Buffer handles most of the material/inventory planning functionality.

It can model all flow of a particular Item, or a subset of that. In modeling a supply chain, a Buffer typically only models the flow of Items at a particular Location (a SKU). Items at different Locations are usually not interchangeable (transportation is needed).

Smaller subsets can be defined. One Buffer could model Items at a Location to be supplied to a very important customer, while another models that Item supplied to other customers. The safety stock maintained for the special customer cannot be treated as interchangeable with the safety stocks for the lesser customers.

One Buffer could model Items at a Location built a certain way (by US Military-approved Resource), while another Buffer models the rest. Those are surely not interchangeable. In fact, it could be argued that such should be different Items. That is not required, but it is required that their flow is managed by different Buffers.

The Buffer defines interchangeability. All Items managed by a Buffer must be interchangeable.

Although most inventory planning functionality is handled by Buffer, it does not directly address the capacity needed to store the Items: it places Load on a Resource to model that needed storage capacity.

The Buffer provides logic to generate Operations / Demand as the result of the flow in and out of it. It manages safety stocks, safety times, lot sizing, and timing.

In each Site, there is a special uneditable Buffer named [unspecified]. It has the default values for all fields. Since it has the UNMODELED flow\_policy extension, the flow from Operations that use the [unspecified] Buffer will not be modeled.

The model has selectors:  
flow\_policy, stocking\_policy,

The Buffer model has these submodels :  
Buffer\_Problem\_Detector.

Models	Buffer Model
--------	--------------

The Buffer model has fields that references these models :  
Item, Location, Operation, Buffer\_Problem\_Detector, Unit, Buffer\_Plan, Site.

These models have a field that is a Buffer model :  
Buffer\_Plan, LINK, Flow, Buffer\_Problem\_Detector, Flow\_Criterion.

The key field for this model is name  
This model may be extended with user-defined fields.

name - - *a Symbol field of model Buffer*  
The name of this Buffer. This name must be unique among Buffers at this Site.  
Default: None -- this is a key field

description - - *a String field of model Buffer*  
A description of this Buffer.  
Default: none

category - - *a Symbol field of model Buffer*  
A simple user-defined categorization of Buffers. This is often used in reports. This will allow users to filter the Buffers by category, and to write expressions that depend upon the category of the Buffers.

The standard reports look for the values "Inventory", "Phantom", "Capacity", and "Cash". An "Inventory" Buffer contains physical Items, and can usually be further categorized by the Item category. Note that a "Resource" Item may be either in an "Inventory" Buffer or in a "Capacity" Buffer, depending upon the nature of it. A "Phantom" Buffer is in the model structure, but is best treated as if it was not. The Item.category is often "Phantom", but may not be (the Item may not be treated as a "Phantom" by all Buffers). A "Capacity" Buffer models capacity that is flow-limited rather than load-limited (e.g., subcontractor that commits to a certain quantity). The Item.category is usually "Resource". A "Cash" Buffer can be used to model cash flow. The Item.category is usually "Cash".  
Default: Inventory

Item - - *a Item field of model Buffer*  
The Item managed by this Buffer.  
  
If this Item is not 'real' (e.g., (unspecified)), then this Buffer cannot be consumed from or produced into. Such a Buffer is typically only used as a 'family' for other Buffers.  
Default: (unspecified)

Models	Buffer Model
--------	--------------

location - - *a Location field of model Buffer*  
The Location of this Buffer. Note that this is not necessarily where the Items are stored -- that is defined by the storage\_operation. Rather, this is the Location of the logical Flow point (staging area, etc.). Typically, though, the storage\_operation's Location is the same as or within this Location.  
Default: (unspecified)

flow\_policy - - *an extension selector of model Buffer*  
This extension defines how the Operation Flows placed on this Buffer are planned. The associated fields define the Problems that can be caused by the flow of Items through this Buffer, and how those Problems are resolved. In particular, it defines the relationship between consumers from the Buffer and producers into it. For example, Lot-For-Lot, Fixed, EOQ, and POQ inventory policies would be implemented as different flow\_policy's.  
Default: LFL\_SIMPLE

Extensions:  
BUCKETED\_NESTED\_SORT, PRODUCING\_FLOW\_CALENDAR, PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK, SUPPLY\_CALENDAR, ON\_HAND\_CALENDAR, ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK, FLOW\_LIMIT\_CALENDAR, FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK, INFINITE\_SUPPLIER, BASIC\_BASIC\_FILTER\_AND\_RANK, FIXED\_QUANTITY\_MULTIPLE, MULTIPLE\_FILTER\_AND\_RANK, FIXED\_QUANTITY\_FENCED, FIXED\_TIME, LFL\_SIMPLE, LFL\_BOUNDED, MLFL\_BOUNDED,

stocking\_policy - - *an extension selector of model Buffer*  
Specifies formula and/or algorithm to calculate the safety stock (min\_on\_hand and/or min\_time) using local and/or global information on supply, process and demand variability and customer service expectation. Note that the safety stock levels will be enforced by the buffer's flow\_policy.

Note that setting stocking\_policy to anything other than MANUAL is irrelevant for a non-inventory carrying flow\_policy such as LFL\_SIMPLE, SUPPLIER etc.

Currently CALENDAR stocking\_policy is only implemented for the PRODUCING\_FLOW\_CALENDAR flow\_policy. CALENDAR stocking\_policy automatically computes time-phased safety stock. It also computes separate safety stock due to demand and supply variability. Note that the user can either bias or override the automatically computed safety stock by specifying a custom formula. See the CALENDAR stocking\_policy description for more details.

Default: MANUAL  
Extensions:  
MANUAL, CALENDAR.

**problem\_detectors** - - *a list of Buffer\_Problem\_Detector submodels of model Buffer*

A list of additional Problem Detectors that are applied to this Buffer. They are notified on each change in flow and have the opportunity to identify specific Problems.

Such Problems are resolved in a fairly disintegrated fashion compared to a 'flow\_policy' designed to deal with the same issues, but the Problems are accurately identified. Some may require manual aid for intelligent resolution. Some will work well with certain 'flow\_policy's, but not others.

The primary intent is to provide full flexibility and orthogonality in describing all of the Problems that need to be detected, even if an intelligent solver (in the form of a 'flow\_policy' extension) has not been provided for that particular combination of Problems.

For example, there may be no 'flow\_policy' that deals with perishing inventory, minimum safety stock levels, and prioritized allocation decisions. Despite that, we may have individual Buffer\_Problem\_Detectors that can deal with each one of those, individually. By just dropping each of those into this list independently, all of those Problems will be accurately detected, supporting full visibility. However, there would be no algorithm designed for intelligently planning with that specific combination of four 'problem\_detectors'.

**level** - - *a Integer field of model Buffer*

Every Buffer in a Site is assigned a level code signifying the relative level upstream from the end Item Buffers (the Buffers consumed only by delivery Operations). The end Item Buffers are level 0. The Buffers that are consumed by Operations that produce into end Item Buffers are level 1. And so on back through the 'producing\_operations' of the Buffers. The value of this field is computed by traversing the FLO network of the owner site.

Level codes can be used algorithmically (as is done in traditional MRP) to traverse all of the Buffer\_Plans in an orderly way such that all the requirements on a Buffer are computed prior to computing its requirements on other Buffers.

Level codes are also very useful in displaying information about the Buffers. By laying out the Buffers graphically according to the level codes, the flow of material can be kept in mostly one direction.

Note that unlike the traditional definition of level codes, the 'level' values must accommodate cyclic material flow (for example, consider a mold that is consumed by an Operation, flows with the material in the routing until it is later removed, cleaned, and then returned to the Buffer from where it started). The 'level' values of Buffers in a cycle are computed from the Buffer with largest 'level' value that is produced from any of the Buffers in the cycle.

Buffers that are not consumed by the producing\_operation of any other Buffer will be assigned level 0.

Properties: command=True Export-Only Field

**all\_producing\_operations** - - *a List(Operation) field of model Buffer*

A where-used analysis that returns all Operations that have a Flow that produces into this Buffer. Note that this is a Read-Only field that has no effect on the planning of this Buffer. (In contrast, the field 'producing\_operation' is commonly used by 'flow\_policy' extensions to specify which Operation should be planned to supply it when needed.)

Properties: Export-Only Field

**all\_consuming\_operations** - - *a List(Operation) field of model Buffer*

A where-used analysis that returns all Operations that have a Flow that consumes from this Buffer. Note that this is a Read-Only field that has no effect on the planning of this Buffer. (In contrast, the field 'consuming\_operation' is commonly used by 'flow\_policy' extensions to specify which Operation should be planned to consume from it when needed.)

Properties: Export-Only Field

**unit** - - *a Unit field of model Buffer*

The 'unit' defines the Quantity of one unit of this Buffer, whether this Buffer is 'discrete' (can only hold Items in whole units), and what the 'preferred\_measure' for this Buffer is. It can also 'convert' between different units of Measure and units of this Buffer.

For example, one unit of Buffer X may be "3 kg", "400 ml", and "\$12". It may be 'discrete', meaning that a need for "5 kg" will result in "6 kg" (2 units) being built. The 'preferred\_measure' may be "kg" which will cause most fields that display a Quantity of this Buffer to display in "kg" by default.

Note that this field is not settable -- you cannot assign it a new Unit. Rather, you can get the Unit of this Buffer and set that Unit's fields.

Properties: Export-Only Field



Models	Buffer Model
--------	--------------

buffer\_plan (Plan) - - a Buffer\_Plan field of model Buffer  
 Returns the Buffer\_Plan for this Buffer in the given Plan.  
 Properties:    Export-Only Field

all\_supplying\_operations - - a List(Operation) field of model Buffer  
 Obsolete! This field has been renamed 'all\_producing\_operations' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.  
 Properties:    obsolete=True    Export-Only Field

owner - - a Site field of model Buffer  
 The Site to which this Buffer belongs.  
 Properties:    Export-Only Field

Models	Buffer_Problem_Detector Model
--------	-------------------------------

### 5.1.15.1 Buffer\_Problem\_Detector Model

Buffer\_Problem\_Detector -- a submodel of model Buffer

A list of additional Problem Detectors that are applied to this Buffer. They are notified on each change in flow and have the opportunity to identify specific Problems.

Such Problems are resolved in a fairly disintegrated fashion compared to a 'flow\_policy' designed to deal with the same issues, but the Problems are accurately identified. Some may require manual aid for intelligent resolution. Some will work well with certain 'flow\_policy's, but not others.

The primary intent is to provide full flexibility and orthogonality in describing all of the Problems that need to be detected, even if an intelligent solver (in the form of a 'flow\_policy' extension) has not been provided for that particular combination of Problems.

For example, there may be no 'flow\_policy' that deals with perishing inventory, minimum safety stock levels, and prioritized allocation decisions. Despite that, we may have individual Buffer\_Problem\_Detectors that can deal with each one of those, individually. By just dropping each of those into this list independently, all of those Problems will be accurately detected, supporting full visibility. However, there would be no algorithm designed for intelligently planning with that specific combination of four 'problem\_detectors'.

The model has selectors:  
 detector.

The Buffer\_Problem\_Detector model has fields that references these models :  
 Buffer.

These models have a field that is a Buffer\_Problem\_Detector model :  
 Buffer.

The key field for this model is detector

detector - - an extension selector of model Buffer\_Problem\_Detector  
 The problem to be identified, and possibly resolved.  
 Default:    None -- this is a key field  
 Extensions:

Models	Buffer_Problem_Detector Model
--------	-------------------------------

feasible - - *a Logical field of model Buffer\_Problem\_Detector*  
 If "false", then this Problem represents an infeasibility that must be resolved to make the plan feasible. In that case, this will have infinite cost.

Setting this from "true" to "false" also sets cost to "00". Setting this from "false" to "true" also sets cost from "00" to "0". Note that this field is strictly unnecessary -- cost is sufficient -- this is provided purely for convenience.

Default: false  
 Properties: command=True

cost - - *a Money field of model Buffer\_Problem\_Detector*

The cost incurred if this Problem is not resolved. If infinite, then this Problem represents an infeasibility that must be resolved to make the plan feasible. In that case, the field 'feasible' will be "false".

Setting this to "00" also sets 'feasible' to "false". Setting this to a finite value also sets 'feasible' to "true".  
 Default: 00

owner - - *a Buffer field of model Buffer\_Problem\_Detector*

Properties: Export-Only Field

Models	flow_policy submodels of model Buffer
--------	---------------------------------------

### 5.1.1.5.2 flow\_policy submodels of model Buffer

#### 5.1.1.5.3 Flow\_Criterion Model

Flow\_Criterion - - *a submodel of model Buffer*

The criterion and relative rank for judging the rank of the Flow\_Plan. It generally corresponds to one of the many properties of the Flow\_Plan, its Operation\_Plan, or the Delivery\_Request or Delivery\_Promise that motivated it.

The model has selectors:  
 criterion.

The Flow\_Criterion model has fields that references these models :  
 Buffer.

These models have a field that is a Flow\_Criterion model :  
 BUCKETED\_NESTED\_SORT, BUCKETED\_COMBINED\_SORT.

The key field for this model is criterion

criterion - - *an extension selector of model Flow\_Criterion*

The criterion for judging the rank of the Flow\_Plan. It generally corresponds to one of the many properties of the Flow\_Plan, its Operation\_Plan, or the Delivery\_Request or Delivery\_Promise that motivated it.  
 Default: None -- this is a key field

Extensions:  
 CUSTOMER\_RANK, SELLER\_RANK, REQUEST\_RANK,  
 ACTUAL\_OR\_FORECAST, REQUEST\_ISSUED, PROMISE\_DUE,  
 REQUEST\_DUE, DUE\_DATE, PROMISED, ENTRY\_DATE,

rank - - *a Number field of model Flow\_Criterion*  
 The importance, weight, or rank of this criterion relative to the others. Some policies may sort by it, others may mathematically weight by it.

Note, as with all rank's, the higher the Number, the higher the rank, the greater the importance, the larger the weight.  
 Default: 0

owner - - *a Buffer field of model Flow\_Criterion*

Properties: Export-Only Field

Models	Resource Model
--------	----------------

### 5.1.1.6 Resource Model

Resource -- *a submodel of model Site*

A Resource models the capacity to perform Operations. This includes machines, tools, fixtures, labor, trucks, molds, dies, masks, and other things that are used by Operations in causing Flow between Buffers. It also includes storage space, containers, racks, and other things that are used to hold Items within Buffers or during Operations.

Fundamentally, a Resource provides a certain efficiency level that may change over time, which defines its basic capacity. It also has a variability level that defines how to compensate the plans to prevent the variability from invalidating them. It may have setup Operations imposed between normal Operations, and maintenance Operations at fixed or computed dates.

The intelligent automated planning of the Loads on the Resource is done as specified by the Resource's load\_policy, which can impose various rules, restrictions, and conditions on the loading of the Resource. In addition, for maximum user flexibility, there is an extra list of Resource\_Problem\_Detectors that can each impose certain restrictions or conditions on the Resource.

In each Site, there is a special uneditable Resource named [unspecified]. It is the root 'family' to which all Resources of the 'owner' Site belong, directly or indirectly. It has the default values for all fields. Since it has the INFINITE load\_policy extension, the load from Operations that use the [unspecified] Resource is ignored.

The model has selectors:  
efficiency, variability, maintenance, size, load\_policy,

The Resource model has these submodels :  
Resource\_Skill, Resource\_Problem\_Detector.

The Resource model has fields that references these models :  
Location, Resource\_Skill, Operation, Resource\_Problem\_Detector,  
Resource\_Plan, Site.

These models have a field that is a Resource model :  
Resource\_Plan, LINK, Load, Resource\_Skill,  
Resource\_Problem\_Detector, Skill\_Resource, MPPS\_Operation\_Resource,  
MPPS\_Batch, PRIMARY, PREFER\_PRIMARY, Size\_Dimension.

Models	Resource Model
--------	----------------

The key field for this model is name  
This model may be extended with user-defined fields.

name -- *a Symbol field of model Resource*  
The name of this Resource. This 'name' must be unique among Resources at this Site.  
Default: None -- this is a key field

description -- *a String field of model Resource*  
A description of this Resource.  
Default: none

category -- *a Symbol field of model Resource*  
A simple user-defined categorization of Resources. This is often used in reports. This will allow users to filter the Resources by category, and to write expressions that depend upon the category of the Resources.

The standard reports look for the values "Labor", "Manufacturing", "Transportation", "Tool", "Fixture", "Rework", "Setup", and "Maintenance".  
Default: Machine

sub\_category -- *a Symbol field of model Resource*  
A simple user-defined categorization of Resources. This is often used in reports. This will allow users to filter the Resources by category, and to write expressions that depend upon the category of the Resources.

The standard reports don not look for any specific values. It is commonly used to categorize by mode, method, duration, cost, or usage pattern. For example, in Transportation, this may be "Express", "Air", "Rail", "Truck", or "Vessel". But any values can be used -- whatever is useful for categorization in reports.  
Default: [unspecified]

location -- *a Location field of model Resource*  
The Location of this Resource, if restricted or fixed. Note that even a movable Resource may specify a Location here that restricts the Resource to Locations within it. For example, a forklift may be able to move to any of 12 docks in Building One, but cannot move to the docks in Building Two. Thus, its 'location' may be 'Building One'. If [unspecified], then the Resource is allowed to move anywhere.  
Default: [unspecified]

Models	Resource Model
--------	----------------

skills - - *a list of Resource\_Skill submodels of model Resource*

The Skills of which this Resource is capable, and its efficiency at those Skills. Note that if the Resource's efficiency at a certain Skill is specified 80%, but the Resource is planned to be operating at 50% efficiency on a Date, then its planned efficiency at that Skill on that Date is only 40% (50% of 80%).

efficiency - - *an extension selector of model Resource*

Defines how the availability, capacity, and efficiency of this Resource is modeled. The associated fields describe the efficiency of this Resource over time, as well as policies for changing/increasing the efficiency.

The efficiency indicates how well the Resource is performing. It affects the time required by Operations: the standard time (std\_time) is based on 100% efficiency. The actual time (time) required for an Operation is the standard time divided by the current efficiency level.

Note that efficiency greater than 100% is legal and supported. For example, a newer machine may operate much faster than the older machines; rather than changing all the Operation times, the new machine can be modeled as having higher than standard (100%) efficiency.

A 0% efficiency effectively indicates unavailability. Thus, the efficiency fields define availability: any time with 0% efficiency is unavailable time; the remainder is the available.

As traditionally defined, capacity is the available hours multiplied by the efficiency during those hours. Thus, the efficiency fields fully define the capacity of this Resource. Note that the capacity of a Resource during a Date\_Range is the amount of load in standard time that can be handled; the available time of a Resource during a Date\_Range is the amount of load in actual time that can be handled.

For example, if a Resource has an efficiency of 50% from 8am to 6pm each day, and 0% the rest of the day, then it is available 10 hours per day, and has a capacity of 5 standard hours each day. If an Operation that takes 3 standard hours is loaded on the Resource, then it will take 6 hours of time (3hr / 50%). The utilization of the Resource can be computed as either 3 / 5 in standard hours, or 6 / 10 in actual hours (60% either way).

Machines typically have continuous availability. Operators, human resources, typically work shifts. However, if operators are not explicitly modeled, a work center may be modeled as working shifts, to reflect the availability of operators to run it.

Models	Resource Model
--------	----------------

Efficiency may be used to reflect regular downtime. For instance, if a machine typically goes down for 6 minutes out of every hour, a convenient way to account for that lost capacity would be to reduce efficiency by 10% (6/60), to 90%. For modelling less regular downtime, see the 'variability' extension.

Efficiency may drop with total load time, until a maintenance Operation is performed to restore it to full performance. Or, efficiency may gradually ramp up over time to reflect a learning curve as a new Resource is being mastered. Or, efficiency may be adjusted in a shift calendar to reflect lower efficiency on night or weekend shifts. And so on.

Default: FIXED, which defaults to 100% efficiency all day, every day

Extensions:

FIXED, CALENDAR,

variability - - *an extension selector of model Resource*

Defines how the expected variability of this Resource is modeled and compensated. A common source of variability is irregular, but not uncommon, failures. Thus, the associated fields may describe frequency of failure (e.g., MTBF -- mean time between failures) and the expected downtime due to each failure (e.g., MTTTR -- mean time to repair).

There are many other sources of variability as well. Variability of efficiency or availability is quite common. If the efficiency can vary from 20% to 90%, but it averages 85%, then how should it be modeled?

Modelling at 85% efficiency is not quite satisfactory -- there will often be times that this is not achieved. Thus, plans based on that will often be infeasible. Modelling at other than 85% (the average) will definitely, over the long term, either under- or over-estimate the available capacity.

The right answer is to model at 85% efficiency, but compensate for the variability by buffering between the Operations on this Resource and downstream Operations by enough time to cover under-performance. In that way, the overall efficiency is modeled properly, at the average, but the plan is buffered such that under-performance does not invalidate it.

Thus, the primary job of the fields associated with the 'variability' extension is to define the time padding added before and after the Operations on this Resource to properly compensate for the variability (prevent the variability from adversely affecting the feasibility of the plans).

Model	Resource Model
-------	----------------

An increase in padding is accompanied by an equivalent increase in WIP for routings that utilize this Resource. Thus, the pads should generally be kept as small as possible without making the plan too brittle in the face of disturbances.

Note that the WIP created by these pads may not sit around this Resource. If on the floor, the next Operation on the dispatch list is always performed as soon as it is available, then the WIP will naturally tend to migrate to in front of the constraint Resource(s) downstream from this Resource. Note that this is the desired behavior: the primary purpose of this padding is to protect the constraint Resource(s) from becoming under-utilized due to the variability of this Resource.

Default: ZERO

Extensions:

ZERO, FIXED,

**maintenance** - - *an extension selector of model Resource*

Defines how maintenance is specified for this Resource. The associated fields describe when maintenance should be performed and what Operation is used to perform it. It may describe both major and minor maintenance Operations. It may base timing either total time, loaded time, setup time, regular calendar intervals, or many other criteria. The Operations may use particular other Resources, which may be available only at certain dates.

Default: ZERO

Extensions:

ZERO,

**size** - - *an extension selector of model Resource*

Defines the size limits on the loads that can be placed on this Resource. The associated fields describe the size (volume, weight) limits of this Resource over time. The size fields indicate how much Load can be handled at once. For example, if the size limit is "2 tons", then up to 2 tons of Loads may be simultaneously processed.

Note that a unitless Quantity as a size typically indicates the ability to process that number of Loads at a time (effectively N identical resources). Thus, a very common size is simply "1". (The default size is the extension "FIXED" which defaults to "1".)

Note that the ability to place multiple loads on a Resource at the same time is defined by the 'load\_policy'. For example, SHARED\_USE and SIMULTANEOUS\_USE allow multiple loads; whereas, EXCLUSIVE\_USE does not. Given a SHARED\_USE load\_policy Resource, the size fields limit how much can be loaded at once.

Default: UNLIMITED

Extensions:

Model	Resource Model
-------	----------------

UNLIMITED, FIXED\_COUNT, FIXED\_QUANTITY, CALENDAR\_COUNT, MULTI\_DIMENSION,

**load\_policy** - - *an extension selector of model Resource*

Defines how the Operation Loads placed on this Resource are planned. The associated fields describe the rules and restrictions on the Loads and the Problems that could occur due to usage of this Resource's capacity. Although many of the Resource characteristics are modeled independent of this extension, any intelligent algorithms to deal with those characteristics will be specified here.

For example, the size limits of this Resource are specified by the 'size' extension and related fields; however, the constraints on shared usage of that size and how to intelligently plan the Operation Loads in order to maximize the use of that size are specified by the 'load\_policy' and related fields.

Thus, the load\_policy extensions selected may depend upon the other extensions selected.

Default: INFINITE

Extensions:

SIMPLE\_CONSOLIDATION, INFINITE\_USE, EXCLUSIVE\_USE,

SHARED\_USE,

**problem\_detectors** - - *a list of Resource\_Problem\_Detector submodels of model Resource*

Do not use Resource\_Problem\_Detectors if there is a 'load\_policy' extension that can be used to model the Resource.

This is a list of additional Problem Detectors that are applied to this Resource. They are notified on each change in load or capacity and have the opportunity to identify specific Problems.

Such Problems are resolved in a fairly disintegrated fashion compared to a 'load\_policy' designed to deal with the same issues, but the Problems are accurately identified. Some may require manual aid for intelligent resolution. Some will work well with certain load\_policy's, but not others.

The primary intent is to provide full flexibility and orthogonality in describing all of the Problems that need to be detected, even if an intelligent solver (in the form of a 'load\_policy' extension) has not been provided for that particular combination of Problems.

Models	Resource Model
--------	----------------

For example, there may be no 'load\_policy' that deals with sequence-dependent setup, maintains a minimum utilization, disallows "green" jobs on Mondays, and prevents four related Resources from all being loaded at once. Despite that, we may have individual Resource\_Problem\_Detectors that can deal with each one of those, individually. By just dropping each of those into this list independently, all of those Problems will be accurately detected, supporting full visibility. However, there would be no algorithm designed for intelligently planning with that specific combination of four 'problem\_detectors'.

**resource\_plan (Plan)** -- *a Resource\_Plan field of model Resource*  
Returns the Resource\_Plan for this Resource in the given Plan.  
Properties:    Export-Only Field

**owner** -- *a Site field of model Resource*  
The Site to which this Resource belongs.  
Properties:    Export-Only Field

Models	Resource_Skill Model
--------	----------------------

### 5.1.1.6.1 Resource\_Skill Model

**Resource\_Skill** -- *a submodel of model Resource*

The Skills of which this Resource is capable, and its efficiency at those Skills. Note that if the Resource's efficiency at a certain Skill is specified 80%, but the Resource is planned to be operating at 50% efficiency on a Date, then its planned efficiency at that Skill on that Date is only 40% (50% of 80%).

The model has selectors:  
efficiency.

The Resource\_Skill model has fields that references these models :  
Skill, Resource.

These models have a field that is a Resource\_Skill model :  
Resource.

The key field for this model is skill

**skill** -- *a Skill field of model Resource\_Skill*  
A skill of which this Resource is capable.  
Default:    None -- this is a key field

**efficiency\_level (Date)** -- *a Percentage field of model Resource\_Skill*  
The efficiency of the 'owner' Resource in performing the 'skill' on the given Date. The efficiency may vary with Date, as specified by the 'efficiency' extension.  
Properties:    Export-Only Field

**efficiency** -- *an extension selector of model Resource\_Skill*  
Specifies the efficiency of the 'owner' Resource in performing 'skill'. Various curves are possible, supporting learning curves, ramp-up, and any other variation with Date. The efficiency curve can be FIXED, or from a CALENDAR.  
Default:    FIXED  
Extensions:  
FIXED, CALENDAR.

**owner** -- *a Resource field of model Resource\_Skill*  
The Resource that is capable of this 'skill' with this 'efficiency'.  
Properties:    Export-Only Field

Models	efficiency submodels of model Resource
--------	--

5.1.1.6.2 efficiency submodels of model Resource

Models	size submodels of model Resource
--------	----------------------------------

5.1.1.6.3 size submodels of model Resource

5.1.1.6.4 Size\_Dimension Model

Size\_Dimension -- a submodel of model Resource

The various dimensions of the size of each one of the aggregate resource. The dimensions need not be orthogonal.

The Size\_Dimension model has fields that references these models : Resource.

These models have a field that is a Size\_Dimension model : MULTI\_DIMENSION.

The key field for this model is name

name -- a Symbol field of model Size\_Dimension  
The name of this dimension of the size  
Default: None -- this is a key field

min\_size -- a Quantity field of model Size\_Dimension  
The minimum total size of Operation\_Plans that can run on this Resource at once.  
Default is 0.  
Default: 0

max\_size -- a Quantity field of model Size\_Dimension.  
The maximum total size of Operation\_Plans that can run on this Resource at once.  
Default is infinite.  
Default: oo (infinite, unlimited, unless)

owner -- a Resource field of model Size\_Dimension

Properties: Export-Only Field

5.1.1.6.5 load\_policy submodels of model Resource

5.1.1.6.6 Resource\_Setup\_Order Model

Resource\_Setup\_Order -- a submodel of model Resource

These models have a field that is a Resource\_Setup\_Order model :  
SHARED\_USE\_SDS, SIMULTANEOUS\_USE\_SDS.

5.1.1.6.7 Resource\_Blocks Model

Resource\_Blocks -- a submodel of model Resource

These models have a field that is a Resource\_Blocks model :  
BLOCK\_CALENDAR, BLOCK\_CYCLE.

5.1.1.7 Skill Model

Skill -- a submodel of model Site

A Skill models a basic capability needed by a resource to perform an Operation. Different Resources may be capable of the same Skill, but possibly at different efficiencies. An Operation can specify a Skill, then during the planning, any Resource capable of that Skill can be used. All the resources within one skill group are considered alternatives of each other.

Note that the (unspecified) Resource results in a similarly special, uneditable Skill named (unspecified) which contains that one (unspecified) Resource with efficiency "100%".

The model has selectors:  
selection.

The Skill model has these submodels :  
Skill\_Resource.

The Skill model has fields that references these models :  
Skill\_Resource, Site.

These models have a field that is a Skill model :  
Resource\_Plan, LINK, Load, SAME\_RESOURCE, Resource\_Skill, Skill\_Resource.

The key field for this model is name  
This model may be extended with user-defined fields.

name -- a Symbol field of model Skill  
The name of this Skill. This 'name' must be unique among Skills at this Site.  
Default: None -- this is a key field

description -- a String field of model Skill  
A description of this Skill.  
Default: none

resources -- a list of Skill\_Resource submodels of model Skill  
The Resources that have this Skill, and their efficiencies at this Skill.

selection -- an extension selector of model Skill

Describes the logic to be used to select one of the 'resources' from all the alternate resources within this skill group. For example, MAX\_EFFICIENCY selects a resource among alternates based on the highest efficiency.

Default: MAX\_EFFICIENCY  
Extensions:  
PRIMARY, PREFER\_PRIMARY, EVEN, MAX\_EFFICIENCY,

loading\_operations -- a List(Operation) field of model Skill  
A where-used analysis that returns all Operations that need this Skill.

Properties: Export-Only Field

loading\_buffers -- a List(Buffer) field of model Skill

A where-used analysis that returns all Buffers that has supplying, receiving, storage or picking operation that needs this Skill.

Properties: Export-Only Field

owner -- a Site field of model Skill

Properties: Export-Only Field



### 5.1.1.7.1 Skill\_Resource Model

Skill\_Resource -- a submodel of model Skill

The Resources that have this Skill, and their efficiencies at this Skill.

The model has selectors:  
efficiency.

The Skill\_Resource model has fields that references these models :  
Resource, Skill.

These models have a field that is a Skill\_Resource model :  
Skill.

The key field for this model is resource

resource -- a Resource field of model Skill\_Resource  
A Resource capable of the owner Skill.

Default: None -- this is a key field

efficiency\_level (Date) -- a Percentage field of model Skill\_Resource  
The efficiency of 'resource' in performing 'owner' Skill on the given Date. The effi-  
ciency may vary with Date, as specified by the 'efficiency'.  
Properties: Export-Only Field

efficiency\_profile (Date\_Range) -- a List(Profile\_Percentage) field of model  
Skill\_Resource

A List of all changes in the efficiency profile during the specified finite Date\_Range.  
This efficiency changes are specific to 'resource' while performing the owner skill.  
Properties: Export-Only Field

efficiency -- an extension selector of model Skill\_Resource  
Specifies the efficiency of 'resource' in performing the 'owner' Skill. Various curves are  
possible, supporting learning curves, ramp-up, and any other variation with Date. The  
efficiency can be FIXED or from a CALENDAR.  
Default: FIXED  
Extensions:  
FIXED, CALENDAR.

owner -- a Skill field of model Skill\_Resource

The Skill that this 'resource' is capable with this 'efficiency'.

Properties: Export-Only Field

### 5.1.1.8 Operation Model

Operation -- a submodel of model Site

Operation Definition

Operation is a submodel of the Site model. An Operation models a  
process, activity, or action that transforms or moves Items. This  
results in a flow into, out of, or between buffers.

Operations may required Resources with specific Skills, modeled  
by Loads. Those Resources model the capacity to perform  
Operations. Flows model the flow of Items to and from Buffers that  
result from Operations.

An Operation has a "unit" of its own, which can be specified in one or more Measures.  
Thus, one unit of an Operation may be 5 liters, 8.78 kg, and \$36.50.  
All proportional Times and Quantities associated with an Operation are relative to  
that unit. The proportional Quantities generally have the suffix \_per  
meaning per unit of the Operation.

Operations can be defined to be discrete. This means they cannot  
run in fractional units.

Example

If the Flows define that 8 legs and 2 tops produce 2 tables, then one unit  
of the Operation produces 2 tables. The times that are given in  
the 'process' extension will be for one unit, or 2 tables. So, if the run\_time is 1 hour  
per unit, then it takes 1 hour to produce 2 tables.

Operation Model Behavior

An Operation can use any number of Resources, with different run\_times, and  
with staggered start and end dates. Therefore, a single activity or a whole series of  
activities can be modeled with a single Operation. This is important for having suffi-  
cient  
flexibility to model the diverse activities in different manufacturers, even within a

single supply chain. It is also important for the planner/modeler to understand, so that it can be used to maximum advantage when deciding what to model and how to model it.

In particular, note that an Operation can model a complete routing. A routing transforms or moves Items and it uses Resources to do it, which is the definition of an Operation. Further, with the extensible model architecture, the behavior of an Operation can be defined in many different ways.

In particular, an Operation can be defined in terms of other Operations (e.g. a routing).

An Operation can consist of other Operations that are run in sequence. The relationships between those sub\_operations can be different depending upon the chosen extension. So, an Operation can model a simple routing (a sequence of Operations allowed to spread), a flow routing (a sequence of Operations which must flow into one another), a set of Operations that can be run at the same time, alternates (a set of alternate Operations), and other combinations (see the process extension for examples).

An Operation can also model simply the bill-of-materials for an Item (Operations that just model the transform of Items, but do not model the capacity required to do that). Similarly, an Operation can model simply the bill-of-distribution for an Item (Operations that just model the movement of Items between Buffers, but do not model the capacity required to do that). Other Operations can then combine the bill-of-materials or bill-of-distribution Operations with Operations that properly consume capacity. Such separation is common in older manufacturing databases.

Note that some Operation specifications could define multiple Operations. For example, some manufacturing software defines Operations such that Resources can be loaded during portions of the Operation time. For ease of interfacing with such packages or databases, process extensions can be provided with the identical specification.

Example

You may have a simple routing Operation that consists of five Operations. One of those five may be an alternates Operation, which consists of four possible Operations.

One of those may itself be a simple routing Operation consisting of three Operations. Therefore, depending upon which alternate is chosen, you may have five Operations or seven Operations to perform.

Advantage of Operation Model Design

The advantage of the Operation model design is simplicity and consistency. By providing a simple building block and the flexibility to combine those blocks, a great deal of modeling capability is provided without complicating the common, simple cases. In this way, the critical Operations and Resources can be modeled in adequate detail while the non-critical models remain suitably inexpensive.

Submodels

The Operation Model consists of two submodels that are essential to its definition: the Flow and Load submodels. Complete understanding of the Operation Model cannot be achieved without understanding those submodels.

Submodel: Flow

The Flow model defines the consumption of Items from a Buffer or the production of Items to a Buffer. The Flow model contains the bill-of-material information. Each Flow can define an independent usage\_policy that defines how much is consumed or produced per unit of the Operation and any yield information. The Flow can also define transfer batches which cause the material to flow at the rate of the Operation (rather than in one lump). Generally, consuming Flow-s occur at the beginning of the Operation: the first transfer batch flows at the start of the Operation and the rest follows at the rate of the Operation until the full Flow has been transferred. Similarly, producing Flow-s generally occur at the end of the Operation: a full transfer batch will occur at the end of the Operation and the rest will be transferred at the rate of the Operation preceding the end such that the full Flow is transferred. That pattern is common, but process extensions can define alternative Flow patterns. In particular, many fixed-time processes do not have a rate, and thus transfer the full consuming Flow at the start and the full producing Flow at the end.

Models	Operation Model
--------	-----------------

See the Flow model for further detail.

Submodel: Load

The Load model defines the capacity required to perform the Operation. For each Load model in the Operation model, one or more Resources will be loaded by that Operation simultaneously. Each Load can specify a single Resource to be loaded, or a Skill group made up of numerous Resources. Generally each Load is applied for the full duration of the Operation. However, that can vary with each process extension. Any process extension that applies the Loads differently will describe that in its documentation. See the Load model for further detail.

Operation Model Default Value

In each Site, there is a special uneditable Operation named [unspecified]. It is the root 'family' to which all Operations of the 'owner' Site belong, directly or indirectly. It has the default values for all fields. Since it has no Loads, no Flows, and takes no Time, it is suitable as a do-nothing Operation for the default value of most Operation fields. For example, the default value of the Resource 'setup\_operation' is [unspecified]. The name [do\_nothing] might be more accurate, but would be inconsistent with the many other [unspecified] models.

Operation Model Selectors

The Operation Model has the following selectors:

process

Operation Model Submodels

The Operation Model has the following submodels:

Flow	Load
Operation_Problem_Detector	
Operation Model Fields	

Models	Operation Model
--------	-----------------

The Operation Model has fields that reference these models:

Flow	Load
Operation_Problem_Detector Site	Unit

Models with Operation Model Field

These models have a field that is an Operation Model:

BUCKETED_COMBINED_SORT	ACCESSORY	Alternate Operation	BASIC
DAR	DUAL_WEIGHTED_NESTED_SORT	Buffer	CALENDAR
Effective_Calendar_Operation	FIXED_QUANTITY_FIXED_QUANTITY_FENCED		
FIXED_TIME	Flow	FLOW_MIN_ON_HAND	
FLOW_MIN_TIME	Flow	FLOW_THRU	Item
LFL_BOUNDED	LFL_MIN_TIME	LFL_SIMPLE	LINK
Load	LOAD_TIME	MLFL_BOUNDED	
MPPS_Operation_Resource	MULTIPLE	Operation_Model_Detector	
Operation_Plan	OPTION	PHANTOM	
PRODUCING_FLOW_CALENDAR	Request_Alternates	Resource	
REWORK	Routing_Operation	SAME_RESOURCE	
Unordered_Operation	WRAPPER		

Operation Model Key Field

The model has selectors:

process,

The Operation model has these submodels :  
Load, Flow, Operation\_Problem\_Detector.

The Operation model has fields that references these models :  
Load, Flow, Operation\_Problem\_Detector, Unit, Site.

These models have a field that is a Operation model :  
Operation\_Plan, Resource, Buffer, LINK, Item, Load, Flow, Operation\_Problem\_Detector, SAME\_RESOURCE, Alternate\_Operation, Effective\_Calendar\_Operation, ACCESSORY, OPTION, EARLIEST, MPPS\_Operation\_Resource, Request\_Alternates, Routing\_Operation, Unordered\_Operation, WRAPPER, REWORK, LOAD\_TIME, DUAL\_WEIGHTED\_TIME, CALENDAR, BUCKETED\_NESTED\_SORT,

Models	Operation Model
--------	-----------------

BUCKETED\_COMBINED\_SORT, CALENDAR,  
PRODUCING\_FLOW\_CALENDAR,  
PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK,  
FLOW\_THRU, FLOW\_MIN\_TIME, FLOW\_MIN\_ON\_HAND, PHAN-  
TOM, BASIC, BASIC\_FILTER\_AND\_RANK, FIXED\_QUANTITY,  
MULTIPLE, MULTIPLE\_FILTER\_AND\_RANK,  
FIXED\_QUANTITY\_FENCED, FIXED\_TIME, LFL\_SIMPLE,  
LFL\_BOUNDED, MLFL\_BOUNDED, LFL\_MIN\_TIME.

The key field for this model is name  
This model may be extended with user-defined fields.

name - - *a Symbol field of model Operation*  
name Field The name of this Operation. This 'name' must be unique among Operations  
at this Site. In many environments this is called the "opcode", "operation code", or  
"operation ID".  
Default: None -- this is a key field

description - - *a String field of model Operation*  
description Field A description of this Operation.  
Default: none

category - - *a Symbol field of model Operation*  
category Field A simple user-defined categorization of Operations. This is often used  
in reports. This will allow users to filter the Operations by category, and to write  
expressions that depend upon the category of the Operations.

The standard reports look for the values "Manufacturing", "Transportation",  
"Rework", "Setup", "Maintenance", "Receiving", and "Delivery".  
Default: Manufacturing

sub\_category - - *a Symbol field of model Operation*  
sub\_category Field A simple user-defined categorization of Operations. This is often  
used in reports. This will allow users to filter the Operations by category, and to write  
expressions that depend upon the category of the Operations.

The standard reports don not look for any specific values. It is commonly used to cate-  
gorize by mode, method, duration, cost, or usage pattern. For example, in Transporta-  
tion, this may be "Express", "Air", "Rail", "Truck", or "Vessel". But any values can be  
used -- whatever is useful for categorization in reports.  
Default: [unspecified]

Models	Operation Model
--------	-----------------

interruptible - - *a Logical field of model Operation*  
interruptible Field If "false", the Operation cannot be planned across breaks, down-  
time, other Operations, or otherwise -- it must be planned in a contiguous block of  
time large enough to complete the Operation.  
Default: false

loads - - *a list of Load submodels of model Operation*  
loads Field Essentially, a Load specifies a Resource that must be loaded to perform the  
Operation. If a Skill is specified by the Load, then a Resource with that Skill must be  
chosen. The Resource will be loaded for the duration of this Operation.

Although each Operation is defined to load the Resources specified by 'loads' through-  
out the duration of the Operation. To model Resources that are loaded for only a por-  
tion of the Operation, you must model the Operation with a 'process' that allows it to  
be described in terms of multiple 'sub\_operations'. For example, a ROUTING Opera-  
tion has 'sub\_operations' which can each load Resources just for their own duration,  
which is a subset of the ROUTING Operation's duration. But Resources specified in  
this 'loads' list for the ROUTING Operation will be loaded for the ROUTING's entire  
duration (for instance, a mold that moves through the whole routing with the material).

Note that this list does not include Loads applied by any 'sub\_operations'. Nor does it  
include the Resources that are used for the duration of any super Operation, which  
would be loaded during this Operation as well.

The efficiency of this Operation is the cumulative product of the efficiencies of the  
Resources that are loaded by this Operation and any super Operations during this  
Operation; or, in the case of all efficiencies being greater than 100%, the minimum of  
those efficiencies is used. Note that the Operation will have one efficiency through-  
out. (\*!\*!\*!\*)

flows - - *a list of Flow submodels of model Operation*  
flows Field The items that will flow from Buffers into this Operation, and the items  
that will flow from this Operation into Buffers. Each may have a different extension:  
some may be discrete, some may be rate-based; some may have yield, some may not;  
etc.

A Flow which specifies an [unspecified] Buffer is considered to an intra-Operation  
Flow used to synchronize upstream and downstream Operations of routings. In partic-  
ular, such a Flow allows you to specify the minimum transfer Quantity into or out of  
this Operation.

Note that this list does not include Flows applied by any 'sub\_operations'.

**consume\_flows** - - *a List(Flow) field of model Operation*  
consume\_flows Field The 'flows' that are being consumed from Buffers by this Operation. To edit the Flows on this Operation, see 'flows'.

Note that this List does not include Flows applied by any 'sub\_operations'. See 'all\_consume\_flows' for a List that includes the Flows of the 'sub\_operations'.  
Properties: Export-Only Field

**produce\_flows** - - *a List(Flow) field of model Operation*  
produce\_flows Field The 'flows' that are being produced to Buffers by this Operation. To edit the Flows on this Operation, see 'flows'.

Note that this List does not include Flows applied by any 'sub\_operations'. See 'all\_produce\_flows' for a List that includes the Flows of the 'sub\_operations'.  
Properties: Export-Only Field

**all\_consume\_flows** - - *a List(Flow) field of model Operation*  
all\_consume\_flows Field The Flows that are being consumed from Buffers by this Operation or any of its 'sub\_operations'. Note that the flows for all alternate Operations will show up, not just the "selected" ones. The "selected" one only has meaning for an Operation\_Plan.  
Properties: Export-Only Field

**all\_produce\_flows** - - *a List(Flow) field of model Operation*  
all\_produce\_flows Field The Flows that are being produced to Buffers by this Operation or any of its 'sub\_operations'. Note that the flows for all alternate Operations will show up, not just the "selected" ones. The "selected" one only has meaning for an Operation\_Plan.  
Properties: Export-Only Field

**all\_flows** - - *a List(Flow) field of model Operation*  
all\_flows Field The Flows that are being produced to and consumed from Buffers by this Operation or any of its 'sub\_operations'. Note that the flows for all alternate Operations will show up, not just the "selected" ones. The "selected" one only has meaning for an Operation\_Plan.  
Properties: Export-Only Field

**sub\_operations** - - *a List(Operation) field of model Operation*  
sub\_operations Field Operations that will be performed as part of this Operation. Every Operation in this List will have this Operation in its 'super\_operations' List.

For example, a ROUTING Operation (see the 'process' extension) will specify a sequence of other Operations that will be performed. Those Operations will appear in this list for analysis purposes. Note that the Operations are not specified here -- they are specified in the fields of the appropriate extension.

Properties: Export-Only Field  
**super\_operations** - - *a List(Operation) field of model Operation*  
super\_operations Field Operations that specify this Operation as a sub\_operation. Every Operation in this List will have this Operation in its 'sub\_operations' List.

For example, if this Operation is a step of a ROUTING Operation (see the 'process' extension), then that ROUTING Operation will appear in this list. Similarly, if this Operation is one of the alternates of an ALTERNATES Operation, then that ALTERNATES Operation will appear in this list.  
Properties: Export-Only Field

**process** - - *an extension selector of model Operation*  
process Field The description of the process performed by this Operation, restrictions on that process, and rules on how to resolve Problems.  
For example, a BASIC Operation (the default) models a time delay that depends upon the number of units of the Operation being performed. The DELAY\_ONLY\_FIXED Operation just models a fixed time delay no matter how many units are involved.

Different processes can define the behavior of the Operation in terms of other Operations. For example, there are various ALTERNATES process extensions which are defined to behave like one of several other Operations. There are various ROUTING process extensions which are defined to perform several other Operations in sequence. Other processes will perform other Operations in an unspecified ordering.  
Default: BASIC

Extensions:  
ALTERNATES\_PRIMARY, ALTERNATES\_PROPORTIONAL,  
EFFECTIVE\_CALENDAR, DELAY\_ONLY\_FIXED, DELAY\_ONLY\_BASIC,  
BASIC\_CALENDARS, FIXED\_TIME, TIME\_MULTIPLE, BASIC,  
BASIC\_DELAYED, REQUEST\_FIXED,  
REQUEST\_FIXED\_WITH\_ANALYSIS, ROUTING,

**problem\_detectors** - - *a list of Operation\_Problem\_Detector submodels of model Operation*  
**operation\_detectors** Field A list of additional Problem Detectors that are applied to this Operation. They are notified on each change to quantity or dates of an Operation\_Plan such that they can identify specific Problems.

Such Problems are resolved in a fairly disintegrated fashion compared to a 'process' designed to deal with the same issues, but the Problems are accurately identified. Some may require manual aid to be resolved well. Some will work well automated with certain processes, but not with others.

The primary intent is to provide full flexibility and orthogonality in describing all of the Problems that need to be detected, even if an intelligent solver (in the form of a 'process' extension) has not been provided for that particular combination of Problems.

For example, there may be no 'process' that .... Despite that, we may have individual Operation\_Problem\_Detectors that can deal with each one of those, individually. By just dropping each of those into this list independently, all of those Problems will be accurately detected, supporting full visibility. However, there would be no algorithm designed for intelligently planning with that specific combination of four 'problem\_detectors'.

**planning\_buffers** - - *a List(Buffer) field of model Operation*  
**planning\_buffers** Field A where-used analysis that returns all Buffers that will generate a plan for this Operation (e.g., to adjust their flow or store their items).  
Properties: Export-Only Field

**planning\_resources** - - *a List(Resource) field of model Operation*  
**planning\_resources** Field A where-used analysis that returns all Resources that will generate a plan for this Operation (e.g., to setup, maintain, or replenish the Resource).  
Properties: Export-Only Field

**operation\_plans** (Plan) - - *a List(Operation\_Plan) field of model Operation*  
**operation\_plans** (Plan) Field Returns a List of the Operation\_Plans of this Operation planned in the specified Plan.  
Properties: Export-Only Field

**unit** - - *a Unit field of model Operation*  
**unit** Field The 'unit' defines the Quantity of one unit of this Operation, whether this Operation is 'discrete' (can only exist in whole units), and what the 'preferred\_measure' for this Item is. It can also 'convert' between different units of Measure and units of this Operation.

For example, one unit of Operation X may be "3 kg", "400 ml", and "\$12". It may be 'discrete', meaning that a need for "5 kg" will result in "6 kg" (2 units) being built. The 'preferred\_measure' may be "kg" which will cause most fields that display a Quantity of this Item to display in "kg" by default.

Note that this field is not settable -- you cannot assign it a new Unit. Rather, you can get the Unit of this Operation and set that Unit's fields.

Default: 0  
Properties: Export-Only Field

**release\_fence** - - *a Horizon\_Date field of model Operation*  
**release\_fence** Field An offset from the "current" date of the plan, by which operation plans should have already been released. Top operation plans with a scheduled start which are not released by the release\_fence will cause a UNRELEASED problem to be raised.

Note that problem detection is performed based solely on the release\_fence of the top operation plan of an operation plan tree; settings for sub\_operation plans are not considered.

Default: 0

**release\_fence\_date** (Plan) - - *a Date field of model Operation*  
**release\_fence\_date** Field Given a Plan, which return the date the Operations release\_fence becomes effective.

Default: 0  
Properties: Export-Only Field

**release\_soon\_fence** - - *a Horizon\_Date field of model Operation*  
**release\_soon\_fence** Field An offset from the "release\_fence" of the operation plan, which serves as warning that the operation plan should be released. Top operation plans with a scheduled start earlier than the release\_soon\_fence will cause a NEEDS\_RELEASE problem to be raised.

Note that problem detection is performed based solely on the release\_soon\_fence of the top operation plan of an operation plan tree; settings for sub\_operation plans are not considered.

Default: 0

Models	Operation Model
--------	-----------------

`release_soon_fence_date (Plan)` -- *a Date field of model Operation*  
`release_soon_fence_date` Field Given a Plan, which return the date the Operations  
`release_soon_fence` becomes effective.

Default: 0  
Properties: Export-Only Field

**release\_name\_expression** - - *a Expression field of model Operation*  
**release\_name\_expression** Field An expression used to calculate the release name to be used for operation plans based on this operation when they are released without specifying a release name. Note that "#" in the expression refers to the Operation\_Plan, not the Operation.

Default: `#.operation.name & "--" & #.operation.next_release_number.string`  
Properties: `command=True`

**next\_release\_number** - *a Integer field of model Operation*  
**next\_release\_number** Field When operation plans are released without specifying a release name, a release name is generated for them, based on an expression (see the release\_name\_expression field). This expression can contain an operation-specific, unique integer so that the release names for all operation plans corresponding to a particular operation are different. next\_release\_number provides the next number in the sequence and is intended for use in release\_name\_expression. Using next\_release\_number in an expression will cause the internal counter to increment each time it is evaluated.

Default: 1  
Properties: command=True Export-Only Field

**release\_number** -- *a Integer field of model Operation*  
**release\_number** Field The last release number generated by next\_release\_number.  
 When the value is zero, no release numbers have yet been generated. The value may  
 be set, in which case the next release number generated with next\_release\_number will  
 be 1 greater than the value set. Note that setting this value smaller than the largest  
 value already used may cause generated names to be duplicated.  
 Default: 0

owner - - a *Site field of model* Operation  
owner Field The Site to which this Operation belongs.  
Properties:    Export-Only Field

**Properties:** **Export-Only Field**

<i>Models</i>	<i>Load Model</i>
---------------	-------------------

#### 5.1.1.8.1 Load Model

### Load -- a submodel of model Operation

The Load model defines the loads put on Resources by the 'owner' Operation. Each Load specifies either a 'resource' or a 'skill' needed to perform the 'owner' Operation, the 'start\_setup' needed to prepare the Resource for the 'owner' Operation, and the 'start\_location' of the 'owner' Operation. It also specifies the 'end\_setup' and 'end\_location' of the Resource following the 'owner' Operation.

Generally, all of the Loads of an Operation are applied simultaneously from the start of the Operation to the end. However, different Operation 'process' extensions can specify that they apply the Loads differently.

The 'usage\_policy' extension of Load specifies how Resources are selected to be loaded. For example, a RESOURCE 'usage\_policy' Load simply uses the one 'resource'; a ONE 'usage\_policy' Load uses one of the Resources capable of the 'skill'; a MAX 'usage\_policy' Load simultaneously uses one or more of the Resources capable of the 'skill', up to a maximum number. In the latter case, how the efficiency, and thus the rate of the Operation, increases with higher numbers of Resources is specified by the fields of the 'usage\_policy'.

Note that even for a ONE Load, a Load that specifies a 'skill' may use multiple Resources during the course of an Operation, just not simultaneously. Consider an Operation which needs a Saw and someone who can operate the Saw. If the Operation will take 3 days, it is unlikely that operator will be willing to work those 3 days straight (the Saw probably won't mind). If the Site runs three shifts, and there are operators in each shift that are capable of operating the Saw, then a ONE Load for that 'skill' will select one operator Resource during each shift. During the 3 days, as many as 9 different Resources could be employed by that single ONE Load.

The model has selectors:  
`usage_policy`.

**The Load model has these submodels :**  
**Load\_Size.**

The Load model has fields that references these models :  
Resource, Skill, Load\_Size, Location, Operation.

These models have a field that is a Load model :  
**Operation**, **Load\_Plan**, **Load\_Size**.

Models	Load Model
--------	------------

The key field for this model is name

name - - *a Symbol field of model Load*

The name for this load. In many cases, a reasonable name for the load is resource.name or skill.name. If the operation puts more than one load on the resource or skill, the names must differ.  
Default: None -- this is a key field

usage\_policy - - *an extension selector of model Load*

Specifies how Resources are loaded by this Operation. The default is RESOURCE, which simply loads the specified 'resource'. Common alternatives are: ONE, which loads one of the Resources with 'skill'; MAX, which loads 'max\_resources', or all available, of Resources with 'skill'; MIN\_MAX, which loads at least 'min\_resources', and up to 'max\_resources' of 'skill'. Each of the latter two allows you to specify a single 'efficiency\_loss' that occurs for each added Resource. Other extensions can provide more precise specifications of efficiency loss, including a table of values for each number of Resources.

Default: RESOURCE

Extensions:

RESOURCE, ONE,

resource - - *a Resource field of model Load*

The Resource required for the 'owner' Operation. This Resource will be loaded for the duration of the Operation.

If the 'usage\_policy' is not RESOURCE, setting this field will also set 'usage\_policy' to RESOURCE and will set the 'skill' field to [unspecified] (you can either specify a Skill or a Resource, but not both).  
Default: [unspecified]

skill - - *a Skill field of model Load*

The Skill required for the 'owner' Operation. A Resource capable of this Skill must be chosen and prepared to perform this Skill. The chosen Resource will be loaded for the duration of the Operation.

If the 'usage\_policy' is RESOURCE, setting this field will also set 'usage\_policy' to ONE and will set the 'resource' field to [unspecified] (you can either specify a Skill or a Resource, but not both).  
Default: [unspecified]

Models	Load Model
--------	------------

load\_sizes - - *a list of Load\_Size submodels of model Load*

Load\_Size represents one dimension of the size of this Load. The Load may have zero, one or more dimensions of size. Each dimension will have a name and a size (in Quantity). For example, a 'box' load may have a "weight" dimension, a "volume" dimension, a "Height" dimension, a "width" dimension and so on. These sizes are used by the resources used by this load.

start\_setup - - *a Symbol field of model Load*

The required starting setup of the Resource for the 'owner' Operation. If none, then no particular start\_setup is required.

Default: none

end\_setup - - *a Symbol field of model Load*

The resulting setup state of the Resource following the 'owner' Operation. If none, then this Operation can leave the Resource in any desired setup. This is only appropriate for setup Operations.

Default: none

start\_location - - *a Location field of model Load*

The required starting Location of the Resource for the 'owner' Operation. If [unspecified], then no particular Location is required.

Default: [unspecified]

end\_location - - *a Location field of model Load*

The resulting Location of the Resource following the 'owner' Operation. If [unspecified], then this Operation can leave the Resource in any Location. That is appropriate for any Fixed-Location Resource, or transit Operations.

Default: [unspecified]

owner - - *a Operation field of model Load*

The Operation that places this Load on a Resource capable of the 'skill'.  
Properties: Export-Only Field



Models	Load_Size Model
--------	-----------------

### 5.1.1.8.1.1 Load\_Size Model

*Load\_Size -- a submodel of model Load*

Load\_Size represents one dimension of the size of this Load. The Load may have zero, one or more dimensions of size. Each dimension will have a name and a size (in Quantity). For example, a box'load may have a "weight" dimension, a "volume" dimension, a "Height" dimension, a "width" dimension and so on. These sizes are used by the resources used by this load.

The model has selectors:

load\_size\_usage.

The Load\_Size model has fields that references these models :  
Load.

These models have a field that is a Load\_Size model :  
Load.

The key field for this model is name

name - - *a Symbol field of model Load\_Size*

The name of this load\_size

Default: None -- this is a key field

load\_size\_usage - - *an extension selector of model Load\_Size*

Represents consumption of the resource in this dimension A FIXED load\_size\_usage always consumes the same 'fixed\_quantity' of the resource irrespective of the units of the op\_plan, whereas a LINEAR load\_size\_usage consumes 'linear\_quantity' of the resource per unit of the op\_plan. The default is LINEAR.

Default: LINEAR

Extensions:

FIXED, LINEAR.

owner - - *a Load field of model Load\_Size*

Properties: Export-Only Field

Models	Flow Model
--------	------------

### 5.1.1.8.2 Flow Model

*Flow -- a submodel of model Operation*

These are the items that are produced or consumed by this Operation. Each may have a different extension -- some may have yield, some may not -- etc. The yield applied to a particular Flow\_Plan is just the yield during the Flow\_Plan, not during the Operation as a whole.

The model has selectors:

usage\_policy.

The Flow model has fields that references these models :  
Buffer, Operation.

These models have a field that is a Flow model :  
Operation, Flow\_Plan.

The key field for this model is name

name - - *a Symbol field of model Flow*

The name for this flow. In many cases, a reasonable name for the load is buffer.name appended onto "PRODUCE" or "CONSUME". If a buffer has more than one PRODUCE or CONSUME flow in the operation, add something to differentiate the names.  
Default: None -- this is a key field

buffer - - *a Buffer field of model Flow*

The Buffer to/from which this Flow of Items will occur.

If the buffer's [unspecified] (or if buffer.phantom is "true"), then this Flow will behave as a phantom Flow (see the 'phantom' field) and not actually be posted to any Buffer.

Default: NULL

phantom - - *a Logical field of model Flow*

If "true", then this Flow is not actually posted to the buffer. Thus, the existence of this Flow is ignored unless used by the super Operation (for example, a ROUTING Operation can use matching phantom Flows between its sub\_operations to determine relative Quantities).

Models	Flow Model
--------	------------

This field is particularly useful when engineering bills or existing bills specify an intermediate item/flow that manufacturing does not want to stock, or where manufacturing has chosen to combine routings together to reduce overhead.

It is also useful for specifying yields for each of the Operations within a routing, even if the intermediate materials are of no interest. In that way, routings can properly compensate for yield loss by increasing the quantity of each upstream sub\_operation.

Note that even if this is "false", if buffer.phantom is "true" or buffer is [unspecified], then this Flow will behave as a phantom Flow.

Refer to the phantom flow\_policy extension of model buffer for more details.  
Default: false

produced - - a Logical field of model Flow  
If "true", then the owner Operation is producing items into the buffer. If "false", then the owner Operation is consuming items from the buffer. This is specified by the flow's usage\_policy and related fields.  
Properties: Export-Only Field

usage\_policy - - an extension selector of model Flow  
Specifies the way this item is consumed/produced by this Operation.  
Default: CONSUME\_PER  
Extensions:  
CONSUME\_FIXED, CONSUME\_PER, PRODUCE\_FIXED, PRODUCE\_PER,  
PRODUCE\_YIELD, PRODUCE\_YIELD\_CALENDAR,  
owner - - a Operation field of model Flow  
Properties: Export-Only Field

Models	usage_policy submodels of model Flow
--------	--------------------------------------

5.1.1.8.2.1 usage\_policy submodels of model Flow

5.1.1.8.3 Operation\_Problem\_Detector Model

Operation\_Problem\_Detector -- a submodel of model Operation

problem\_detectors Field A list of additional Problem Detectors that are applied to this Operation. They are notified on each change to quantity or dates of an Operation\_Plan such that they can identify specific Problems.

Such Problems are resolved in a fairly disintegrated fashion compared to a 'process' designed to deal with the same issues, but the Problems are accurately identified. Some may require manual aid to be resolved well. Some will work well automated with certain 'process's, but not with others.

The primary intent is to provide full flexibility and orthogonality in describing all of the Problems that need to be detected, even if an intelligent solver (in the form of a 'process' extension) has not been provided for that particular combination of Problems.

For example, there may be no 'process' that .... Despite that, we may have individual Operation\_Problem\_Detectors that can deal with each one of those, individually. By just dropping each of those into this list independently, all of those Problems will be accurately detected, supporting full visibility. However, there would be no algorithm designed for intelligently planning with that specific combination of four problem\_detectors.

The model has selectors:  
detector.

The Operation\_Problem\_Detector model has fields that references these models :  
Operation.

These models have a field that is a Operation\_Problem\_Detector model :  
Operation.

The key field for this model is detector

detector - - an extension selector of model Operation\_Problem\_Detector  
The problem to be detected, and possibly resolved.  
Default: None -- this is a key field  
Extensions:

feasible - - a Logical field of model Operation\_Problem\_Detector  
If "false", then this Problem represents an infeasibility that must be resolved to make the plan feasible. In that case, this will have infinite 'cost'.

Setting this from "true" to "false" also sets 'cost' to "oo". Setting this from "false" to "true" also sets 'cost' from "oo" to "0". Note that this field is strictly unnecessary -- cost is sufficient -- this is provided purely for convenience.

Default: false

cost - - a Money field of model Operation\_Problem\_Detector

The cost incurred if this Problem is not resolved. If infinite, then this Problem represents an infeasibility that must be resolved to make the plan feasible. In that case, the field 'feasible' will be "false".

Setting this to "oo" also sets 'feasible' to "false". Setting this to a finite value also sets 'feasible' to "true".

Default: oo

owner - - a Operation field of model Operation\_Problem\_Detector

Properties: Export-Only Field

5.1.1.8.4 process submodels of model Operation  
5.1.1.8.5 Alternate\_Operation Model

Alternate\_Operation -- a submodel of model Operation

One alternate 'operation' that can be selected for performing the 'owner' Operation.

The Alternate\_Operation model has fields that references these models :  
Operation.

These models have a field that is a Alternate\_Operation model :  
ALTERNATES\_PRIMARY, ALTERNATES\_MANUAL,  
ALTERNATES\_PROPORTIONAL, ALTERNATES\_PREFERENCE.

The key field for this model is operation

operation -- a Operation field of model Alternate\_Operation

The Operation to be performed as a 'sub\_operation' of the 'owner' Operation. The same Operation may not appear in the 'owner' Operation multiple times. Operation. Setting this field will cause existing operation plan trees using this alternate to be modified to be consistent with the new value, but the exact effect on those operation plans is undefined. They may be deleted or moved to another alternate.

Default: None -- this is a key field

quantity\_per -- a Quantity field of model Alternate\_Operation

The Quantity of 'operation' that must be performed per unit of the 'owner' Operation. The Quantity is converted into units of 'operation'. And, thus, unless Quantity's, such as the default value of "1", means that number of units of 'operation'. In the case of the default, one unit of 'operation' is performed for each unit of the 'owner' Operation. Setting this field will cause existing operation plan trees using this alternate to be modified to be consistent with the new value, but the exact effect on those operation plans is undefined. They may be resized, deleted, or moved to another alternate.

Default: 1

rank -- a Integer field of model Alternate\_Operation

The rank is used by SEQUENTIAL\_ALTERNATE strategy (see execution extension of strategy). The lower number has higher priority.

Properties: Export-Only Field

percentage -- a Percentage field of model Alternate\_Operation

This may be the percentage of times this alternate 'operation' is typically selected, percentage it should be selected, or the priority of selection.

Default: 100%

owner -- a Operation field of model Alternate\_Operation

The Operation that will perform this 'operation' as one of its 'sub\_operation's.

Properties: Export-Only Field

5.1.1.8.6 Effective\_Calendar\_Operation Model

Effective\_Calendar\_Operation -- a submodel of model Operation

One effective alternate 'operation' that can be selected for performing the 'owner' Operation.

The Effective\_Calendar\_Operation model has fields that references these models :  
Operation, Calendar.

These models have a field that is a Effective\_Calendar\_Operation model :  
EFFECTIVE\_CALENDAR.

The key field for this model is operation

operation -- a Operation field of model Effective\_Calendar\_Operation

The Operation to be performed as a 'sub\_operation' of the 'owner' Operation. The same Operation may not appear in the 'owner' Operation multiple times.

Default: None -- this is a key field

quantity\_per -- a Quantity field of model Effective\_Calendar\_Operation

The Quantity of 'operation' that must be performed per unit of the 'owner' Operation. The Quantity is converted into units of 'operation'. And, thus, unless Quantity's, such as the default value of "1", means that number of units of 'operation'. In the case of the default, one unit of 'operation' is performed for each unit of the 'owner' Operation.

Default: 1

percentage -- a Percentage field of model Effective\_Calendar\_Operation

This may be the percentage of times this alternate 'operation' is typically selected, percentage it should be selected, or the priority of selection.

Default: 100%

calendar -- a Calendar field of model Effective\_Calendar\_Operation

This specifies the effectivity calendar for the 'operation'. Each calendar entry should specify the name of the operation that is effective at that time.

Default: [unspecified]

Models	Routing_Operation Model
--------	-------------------------

owner -- a Operation field of model Effective\_Calendar\_Operation  
 The Operation that will perform this 'operation' as one of its 'sub\_operation's'.  
 Properties: Export-Only Field

### 5.1.1.8.7 Routing\_Operation Model

Routing\_Operation -- a submodel of model Operation

A sequenced sub\_operation of the 'owner' routing Operation.

The Routing\_Operation model has fields that references these models :  
 Operation.

These models have a field that is a Routing\_Operation model :  
 ROUTING, TRANSFER\_ROUTING, FLOW\_ROUTING.

The key field for this model is sequence\_number

sequence\_number -- a Number field of model Routing\_Operation

The smaller the Number, the earlier in the sequence it is placed. The 'sequence\_number's within the 'owner' Operation must be unique.

Default: None -- this is a key field

Properties: required=true

operation -- a Operation field of model Routing\_Operation

The Operation to be performed as a sub\_operation' of the 'owner' Operation. The same Operation may appear in the 'owner' Operation multiple times (with different 'sequence\_number's of course).

Default: [unspecified]

quantity\_per -- a Quantity field of model Routing\_Operation

The Quantity of 'operation' that must be performed per unit of the 'owner' Operation. The Quantity is converted into units of 'operation'. And, thus, unitless Quantity's, such as the default value of "1", means that number of units of 'operation'. In the case of the default, one unit of 'operation' is performed for each unit of the 'owner' Operation.

If this value is less than or equal to 0, then this value is ignored. In that case, the Operation must have a Flow complementary to the successive sub\_operation, and the quantity will be determined based upon that. Given the variety of Flow usage\_policy's, this approach offers much more flexibility. In particular, for modeling yield's and similar complexities. Note that such Flows may be phantom's, or the Buffers they flow to and from may be phantom's; in either case, the Flows will not be posted to a Buffer\_Plan or result in any other planning structures.

Models	Configuration Model
--------	---------------------

Default: 1

owner -- a Operation field of model Routing\_Operation

The Operation that will perform this 'operation' as one of its 'sub\_operation's'.

Properties: Export-Only Field

### 5.1.1.9 Configuration Model

Configuration -- a submodel of model Site

A Configuration models an Item with specific desired characteristics. For example, an OPTIONED Item cannot be built without specifying the desired options. Such information is specified in this, the Configuration.

The model has selectors:  
 spec.

The Configuration model has fields that references these models :  
 Item, Site.

These models have a field that is a Configuration model :

LINK, Item\_Request, Item\_Acceptance, Item\_Promise, Lot, PRODUCE, CONSUME, DELIVER, SIMPLE\_FIXED\_QUANTITY, SINGLE\_REQUEST, SIMPLE\_FIXED\_TIME, WEEKLY, DUAL\_REQUEST, REQUEST\_FIXED, REQUEST\_FIXED\_WITH\_ANALYSIS, Option\_Spec.

The key field for this model is item

item -- a Item field of model Configuration

The Item for which this Configuration specifies desired characteristics.

Default: None -- this is a key field

spec -- an extension selector of model Configuration

The value of this 'spec' extension selector is defined by 'item.spec'. It defines the fields of this Configuration needed to specify the Item precisely enough to define interchangeability.

For instance, an OPTIONED Configuration specifies desired options for various features. The options specified in two Configurations of an OPTIONED Item must match to be interchangeable. A STANDARD Configuration has no specification fields -- they are all interchangeable. In contrast, a CUSTOM Configuration also has no specification fields, but they are all considered distinct (not interchangeable).

Models	Configuration Model
Properties: Export-Only Field Extensions: STANDARD, CUSTOM,	
interchangeable (Configuration) -- a Logical field of model Configuration If "true", then Items of the argument Configuration are interchangeable with Items of this Configuration. They have equivalent characteristics of those that determine interchangeability.	
Properties: Export-Only Field	
owner -- a Site field of model Configuration The Item for which this Configuration specifies the desired characteristics. This is the same as 'Item'.	
Properties: Export-Only Field	

5.1.1.9.1 spec submodels of model Configuration  
5.1.1.10 Item\_Group Model  
Item\_Group -- a submodel of model Site

A Item\_Group models a hierarchical grouping or classification of Items. A Item can only appear directly in only one Item\_Group of a hierarchy. Similarly, a Item\_Group can only appear directly in one place in the hierarchy. However, a Item can appear in any number of separate Item\_Group hierarchies.

An implication of this is that all the Item\_Groups cannot be assembled into the one [unspecified] Item\_Group -- the hierarchies must remain separate, otherwise the totals for one Item would be counted dozens of times or the totals would not add in an understandable way.

For example, milk Items may be grouped by fat content: "skim", "1%", "2%", and "whole". The same milk Items may also be grouped by size: pint, quart, half-gallon, and gallon. Further, the same milk Items may be grouped by brand: "Econo-Cow" and "Pure-White". A particular Item would be in one fat content group, one size group, and one brand group.

Item\_Groups are organized into hierarchies. In the above example, the four fat groups might be in the "Milk by Fat" Item\_Group, the four sizes might be in the "Milk By Size" group, and the two brands might be in the "Milk By Brands" group. Further, if there are non-milk Items, those groups may be included into other Item\_Groups. For instance, "Milk By Size" may be in the "By Drink By Size" Item\_Group, which also includes "Orange\_Juice\_By\_Size" and others.

Note that grouping "Milk by Fat", "Milk by Size", and "Milk by Brands" into a "Milk" group would result in counting each Milk item as many as three times. Each "Milk by \*" group gives the summary of all Milk, but then breaks it out different ways.

All of the above milk Item\_Groups are related to Item characteristics. You can also form Item\_Groups based on other properties of the Items. For instance, your Items may be divided based upon groups of customers (Site\_Groups), or "market channels", or "industry segments". For example, pints of milk sold to grocery stores may be a separate Item from pints of milk sold to restaurants. A Item\_Group may be formed based on that division: "Milk By Channel".

Another common division will be by delivery lead time. Often the same Item will be made available as one Item with a 2-week delivery lead time and another with a 6-week delivery lead time, where the latter has a significantly lower price. When Items are so divided, they will almost always be grouped together for forecast purposes.

The Item\_Group model has these submodels :  
Sub\_Item\_Group, Sub\_Item.

The Item\_Group model has fields that references these models :  
Item\_Group, Sub\_Item\_Group, Site.

These models have a field that is a Item\_Group model :  
LINK, Item\_Group, Sub\_Item\_Group, Sub\_Item.

The key field for this model is name  
This model may be extended with user-defined fields.

name - - a Symbol field of model Item\_Group  
The name of this Item\_Group. This 'name' must be unique among Item\_Groups of the 'owner' Site.

Default: None -- this is a key field

description - - a String field of model Item\_Group  
A description of this Item\_Group.  
Default: none

top - - a Logical field of model Item\_Group  
If "true", then this Item\_Group appears in no other Item\_Groups -- it is the top of a Item\_Group hierarchy.  
Properties: Export-Only Field

root - - a Item\_Group field of model Item\_Group  
The root of this Item\_Group tree.  
Properties: Export-Only Field

sub\_groups - - a list of Sub\_Item\_Group submodels of model Item\_Group  
The Item\_Groups that are direct members of this Item\_Group. Note that no Item\_Group descendant can contain this Item\_Group or contain common Items.

sub\_items - - a list of Sub\_Item submodels of model Item\_Group  
The Items that are direct members of this Item\_Group. This does not include the Items in this Item\_Group's 'sub\_items'. The 'all\_items' field provides a complete list.

all\_items, all\_items (List(Item)) - - a List(Item) field of model Item\_Group  
All of the Items that are in this Item\_Group, both directly and indirectly. This list includes the Items in 'sub\_items' plus 'all\_items' of each of the Item\_Groups in 'sub\_groups'.

If passed List(Item), it returns only those Items that are in the passed List and in this Item\_Group. In other words, it returns the intersection of the Items in this Item\_Group and the argument List.

Properties: Export-Only Field

owner - - a Site field of model Item\_Group  
The Site to which this Item\_Group belongs.  
Properties: Export-Only Field

Models	Sub_Item_Group Model
--------	----------------------

### 5.1.1.10.1 Sub\_Item\_Group Model

#### Sub\_Item\_Group -- a submodel of model Item\_Group

The Item\_Groups that are direct members of this Item\_Group. Note that no Item\_Group descendant can contain this Item\_Group or contain common Items.

The Sub\_Item\_Group model has fields that references these models :  
Item\_Group.

These models have a field that is a Sub\_Item\_Group model :  
Item\_Group.

The key field for this model is item\_group

item\_group -- a Item\_Group field of model Sub\_Item\_Group  
A Item\_Group that is a direct member of the 'owner' Item\_Group.  
Default: None -- this is a key field

owner -- a Item\_Group field of model Sub\_Item\_Group

Properties: Export-Only Field

Models	Sub_Item Model
--------	----------------

### 5.1.1.10.2 Sub\_Item Model

#### Sub\_Item -- a submodel of model Item\_Group

The Items that are direct members of this Item\_Group. This does not include the Items in this Item\_Group's 'sub\_items'. The 'all\_items' field provides a complete list.

The Sub\_Item model has fields that references these models :  
Item, Item\_Group.

These models have a field that is a Sub\_Item model :  
Item\_Group.

The key field for this model is item

item -- a Item field of model Sub\_Item  
A Item that is a direct member of the 'owner' Item\_Group.  
Default: None -- this is a key field

owner -- a Item\_Group field of model Sub\_Item

Properties: Export-Only Field



Models	Seller Model
--------	--------------

### 5.1.2 Seller Model

#### Seller -- a submodel of model Supply\_Chain

A Seller can model a sales person, group, channel, territory, or organization. It represents the responsibility for forecasting demand, committing to sales, managing allocations, taking orders, and promising orders.

Sellers may take Requests from one Site or multiple Sites for items supplied by one Site or several different Sites. It manages the Requests and Promises made between those Sites. A Seller can act as an agent of the supplying Site and make Promises for those Sites.

Sellers can manage Requests and Promises for one Product or many Products. Products can be defined for certain customer(s), certain Item(s), certain Order Lead Time, certain Price, and so on. Each Product can be Forecasted independently, or grouped with other Products into Product\_Groups.

Sellers can form a hierarchy. Each Seller can be a member of another Seller, its 'organization'. Allocations can be made to any level in the hierarchy. Sellers can use allocations to themselves or any of their organizations.

The model has selectors:  
request\_naming.

The Seller model has these submodels :  
Site\_Group, Product\_Root, Product, Product\_Group.

The Seller model has fields that references these models :  
Seller, Site\_Group, Product\_Root, Product\_Group, Horizon, Seller\_Plan, Supply\_Chain.

These models have a field that is a Seller model :  
Supply\_Chain, Seller, Product\_Root, Product, Seller\_Plan, Site\_Group, Product\_Group, Product\_Allocation, REQUEST\_FIXED, REQUEST\_FIXED\_WITH\_ANALYSIS.

The key field for this model is name  
This model may be extended with user-defined fields.

name -- a Symbol field of model Seller  
Full name of this Seller. It must be unique among Sellers in the 'owner' Supply\_Chain.

Models	Seller Model
--------	--------------

Default: None -- this is a key field

description -- a String field of model Seller  
A description of this Seller.  
Default: none

category -- a Symbol field of model Seller  
A simple user-defined categorization of Sellers. This is often used in reports. This will allow users to filter the Sellers by category, and to write expressions that depend upon the category of the Seller.

The standard reports look for the values "Office", "Channel", and "Customer". "Customer" indicates that the Seller models sales broken out by customer. "Channel" indicates that the Seller models sales broken out by market segment. "Office" indicates that the Seller models sales broken out by division, sales office, or store front.  
Default: Channel

sub\_category -- a Symbol field of model Seller  
A simple user-defined categorization of Sellers. This is often used in reports. This will allow users to filter the Sellers by category, and to write expressions that depend upon the category of the Sellers.

The standard reports don not look for any specific values. But any values can be used - whatever is useful for categorization in reports.  
Default: [unspecified]

rank -- a Number field of model Seller  
Rank of this seller. This rank is used to prioritize the allocations. Sellers with higher rank get their allocations before it goes to lower rank sellers. If allocated amount is less than what is needed by sellers it is distributed proportionally among the sellers as specified by MEMBER\_RANK allocation policy. Higher the number, higher the rank is.  
Default: 0

organization -- a Seller field of model Seller  
The Seller organization that this Seller is a member of. A Seller 'organization' can represent a Sales Office, Sales Organization, Sales Manager, Market Channel, Sales Territory, etc.

The Products and Product\_Groups of the 'organization' are used as if they are Products and Product\_Groups of this Seller. This Seller can have Products and Product\_Groups that are not used by its 'organization'.

Models	Seller Model
--------	--------------

Default: [unspecified]

members - - *a List(Seller) field of model Seller*

This List contains each Seller that specifies this Seller as its 'organization'. If this List is empty, then this Seller is not an 'organization' for any other.

Properties: Export-Only Field

member\_of (Seller) - - *a Logical field of model Seller*

Returns "true" if this Seller or any of its 'organization's is the parameter Seller. More precisely, if this Seller is the parameter Seller, then it returns "true"; otherwise, if this Seller's 'organization' is [unspecified], then it returns "false"; otherwise, it returns 'organization.member\_of(parameter)'.  
Properties: Export-Only Field

site\_groups - - *a list of Site\_Group submodels of model Seller*

Each Site\_Group defines a List of Sites. That list can be specified explicitly, or defined by specifying common characteristics (all Sites matching those characteristics are included in the Group). Such matching criteria can use user-defined fields of Site as well, allowing arbitrary grouping and categorization fields to be used with Sites.

Any Site\_Groups defined here will appear in all 'site\_groups' of this Seller and of any of its 'members'.

all\_site\_groups - - *a List(Site\_Group) field of model Seller*

This List of Site\_Groups includes all 'site\_groups' plus the Site\_Groups defined by all of this Seller's 'organization's (e.g., 'organization.organization', etc.). Thus, several Sellers can share Site\_Groups by defining them in a common Seller 'organization'.

If a Seller defines a Site\_Group with the same name as an 'organization's Site\_Group, then the 'organization's Site\_Group will not appear in this List. Only one Site\_Group with a given name will appear, the one defined lowest in the Seller hierarchy. Thus, a Seller can effectively override any Site\_Group of its 'organization'.

This List is not affected by any 'members' of this Seller.

Properties: Export-Only Field

site\_group (Symbol) - - *a Site\_Group field of model Seller*

The Site\_Group defined by this Seller or one of its 'organization's with the specified name.

Models	Seller Model
--------	--------------

Note that 'site\_group(name)' is equivalent to 'all\_site\_groups.find(name)', but is simpler, clearer, and more efficient.

Properties: Export-Only Field

product\_roots - - *a list of Product\_Root submodels of model Seller*

The Product\_Roots defined by this Seller. Each of these defines a Product-Seller tree rooted at this Seller. For each Product\_Root, a Product will be created in the 'products' of this Seller and all of its 'members'. Those Products form the Product-Seller tree.

products - - *a list of Product submodels of model Seller*

The Products sold by this Seller. A Product is created for each Product\_Root in this Seller and each of its 'organization's.

top\_products - - *a List(Product) field of model Seller*

This List is a subset of 'all\_products' consisting only of Products that are the top of a Product hierarchy, the Products that do not appear within any Product\_Group.

Properties: Export-Only Field

active\_products - - *a List(Product) field of model Seller*

This List is a subset of 'all\_products' consisting only of Products that have been 'activated' (see Product.active).

Properties: Export-Only Field

product\_groups - - *a list of Product\_Group submodels of model Seller*

Product\_Groups group Products into hierarchies. Each Product can appear in at most one Product\_Group in a hierarchy of Product\_Groups. But each Product can appear in any number of independent Product\_Group hierarchies.

The purpose of such hierarchical grouping is to allow forecasts to be meaningfully aggregated from Product forecasts and then adjusted and redistributed down the hierarchy to the Products. If the Product\_Groups in a hierarchy contain common Products, then such aggregation is difficult to interpret or redistribute.

Any Product\_Groups defined here will appear in 'product\_groups' of this Seller's 'members'.

top\_product\_groups - - *a List(Product\_Group) field of model Seller*

This List is a subset of 'product\_groups' consisting only of Product\_Groups that are the top of a Product\_Group hierarchy, the Product\_Groups that do not appear within any other Product\_Group.

Properties: Export-Only Field

**forecast\_horizon** - - *a Horizon field of model Seller*

Defines how the Seller\_Plan's forecast\_horizon is computed. This defines the lowest granularity of the forecasts for each of the Products. Each 'member' Seller can define a finer granularity (smaller Date\_Ranges), but not coarser. The forecast\_horizon that results in the Seller\_Plan will include all breaks specified by this horizon definition, as well as all breaks specified by any organization's fields. In a sense, the horizons are laid over the top of each other.  
Default: [unspecified]

**first\_day\_of\_week** - - *a Integer field of model Seller*  
Monday is day 1; Sunday is day 7.

This field is obsolete and no longer be available after the 3.05 release. Use forecast\_horizon.first\_day\_of\_week instead.  
Default: 1  
Properties: obsolete=True

**week\_periods** - - *a Number field of model Seller*  
The number of week periods. After that number of week periods, monthly periods begin. If nonexistent, then the entire horizon is broken into weeks.

This field is obsolete and no longer be available after the 3.05 release. Use forecast\_horizon.week\_periods instead.  
Default: nonexistent  
Properties: obsolete=True

**request\_naming** - - *an extension selector of model Seller*  
Defines how names are generated for the forecast Requests generated by this Seller.  
Default: NUMERIC  
Extensions:  
NONE.

**atp\_horizon** - - *a Horizon field of model Seller*  
Defines how the Seller\_Plan's atp\_horizon is computed. This defines the lowest granularity of the forecasts for each of the Products. Each 'member' Seller can define a finer granularity (smaller Date\_Ranges), but not coarser. The atp\_horizon that results in the Seller\_Plan will include all breaks specified by this horizon definition, as well as all breaks specified by any organization's fields. In a sense, the horizons are laid over the top of each other.  
Default: MONTHS

**seller\_plan (Plan)** - - *a Seller\_Plan field of model Seller*  
The Seller\_Plan for this Seller in the specified Plan.  
Properties: Export-Only Field

**owner** - - *a Supply\_Chain field of model Seller*  
The Supply\_Chain for which this is a Seller.  
Properties: Export-Only Field

5.1.2.1 Site\_Group Model

Site\_Group -- a submodel of model Seller

A Site\_Group models a list of Sites, specified in one of several different ways. It may be specified explicitly by simply naming the Sites that belong in the Site\_Group. Alternatively, it may specify a filter expression which computes whether a Site is included or not. Or it may look for a certain category or set of category values in the Sites.

This is commonly used to specify which Sites are customers of a particular Product.

The model has selectors:  
spec.

The Site\_Group model has these submodels :  
Explicit\_Site.

The Site\_Group model has fields that references these models :  
Explicit\_Site, Seller.

These models have a field that is a Site\_Group model :  
Seller, Product\_Root, Product, Explicit\_Site.

The key field for this model is name  
This model may be extended with user-defined fields.

name - - a Symbol field of model Site\_Group  
The name of this Site\_Group. This 'name' must be unique among the Site\_Groups of the 'owner' Seller.

Default: None -- this is a key field

description - - a String field of model Site\_Group

A description of this Site\_Group.

Default: none

sites, sites (List(Site)) - - a List(Site) field of model Site\_Group

The Sites that are in this Site\_Group. If passed List(Site), it returns only those Sites that are in the passed List and in this Site\_Group. That is, it returns the intersection of the Sites in this Site\_Group and the argument List.

Properties: Export-Only Field

spec - - an extension selector of model Site\_Group

How the list of Sites that make up the Site\_Group are specified. The default is EXPLICIT, where each Site in the group is named explicitly. Future extensions will include filtering extensions that let you filter the sites by 'category', 'name', or any number of other fields, or even an arbitrary Expression which computes whether a Site belongs. Currently, only EXPLICIT is supported.

Default: EXPLICIT

explicit\_sites - - a list of Explicit\_Site submodels of model Site\_Group

Identifies a Site, 'site', that is explicitly included in the 'owner' Site\_Group.

owner - - a Seller field of model Site\_Group  
The Seller to which this Site\_Group belongs.  
Properties: Export-Only Field

Models	Explicit_Site Model
--------	---------------------

### 5.1.2.1.1 Explicit\_Site Model

Explicit\_Site -- *a submodel of model Site\_Group*

Identifies a Site, 'site', that is explicitly included in the 'owner' Site\_Group.

The Explicit\_Site model has fields that references these models :  
Site, Site\_Group.

These models have a field that is a Explicit\_Site model :  
Site\_Group.

The key field for this model is site

site -- *a Site field of model Explicit\_Site*  
The Site that has been explicitly added to this Site\_Group.  
Default: None -- this is a key field

owner -- *a Site\_Group field of model Explicit\_Site*

Properties: Export-Only Field

Models	Product_Root Model
--------	--------------------

### 5.1.2.2 Product\_Root Model

Product\_Root -- *a submodel of model Seller*

Product\_Root defines the base information of a Product. Defining a Product\_Root defines a Product-Seller tree. The root of that tree is the Product in this Seller. The corresponding Products in each of the members of this Seller form the rest of the Product-Seller tree. Products cannot be added directly -- a Product\_Root must be added.

A Product defines one or more Items that are available to a set of customers with a certain delivery lead time to certain delivery territories, at a certain price. Any of those may be unlimited (any delivery lead time; any delivery address; etc.). Each Product is independently forecasted, allocated, and priced for the purposes of quoting/promising. See the Product model for more information on Products.

The Product\_Root defines....

In each Seller, there is a special uneditable Product\_Root named [unspecified]. It has the default values for all fields. Since its Item is [unspecified], this Product is not real and cannot be actually requested, promised, or planned.

The model has selectors:  
forecast\_policy.

The Product\_Root model has these submodels :  
Product\_Supplier.

The Product\_Root model has fields that references these models :  
Product\_Supplier, Site, Site\_Group, Unit, Seller.

These models have a field that is a Product\_Root model :  
Seller, Product, Product\_Supplier, BUCKETED\_NESTED\_SORT.

The key field for this model is name  
This model may be extended with user-defined fields.

Model	Product_Root Model
-------	--------------------

**name** - - *a Symbol field of model Product\_Root*  
 The name of this Product(\_Root). Depending upon the environment, this is typically the name of the Item (Item family, or pseudo item), name of the Buffer (Buffer family, or SKU), or the name of the Operation. However, it is also often none of those. It simply must be unique among the Products of the Owner Seller, but otherwise can be whatever is meaningful.  
 Default: None -- this is a key field

**description** - - *a String field of model Product\_Root*  
 A description of this Product(\_Root).  
 Default: none

**suppliers** - - *a list of Product\_Supplier submodels of model Product\_Root*  
 The Items that are sold as this Product(\_Root), organized by supplying Site. Only Item\_Requests for these Items may be filled by this Product(\_Root). Forecasts for this Product(\_Root) only include sales of these Items.

**min\_quantity** - - *a Quantity field of model Product\_Root*  
 An Item\_Request with a quantity less than this cannot be filled by this Product(\_Root). Forecasts for this Product(\_Root) only include sales with this and larger Quantity. This Quantity is converted to the unit of this Product(\_Root).  
 Default: 0

**min\_delivery\_lead\_time** - - *a Time field of model Product\_Root*  
 An Item\_Request with an delivery lead time less than this cannot be filled by this Product(\_Root). Forecasts for this Product(\_Root) only include sales with this and larger delivery lead time. Delivery lead time is the Time from when the Promise must be accepted (accept\_by) to when the delivery must be made (due\_end).  
 Default: 0

**delivery\_area** - - *a Symbol field of model Product\_Root*  
 Defines the delivery area that this Product(\_Root) covers. Forecasts for this Product(\_Root) only include sales within this delivery area.  
 Default: [unspecified]

**effective\_dates** - - *a Date\_Range field of model Product\_Root*  
 A Request for this Product can be filled during these dates.  
 Default: infinite date range

**customer** - - *a Site field of model Product\_Root*  
 Item\_Requests from this customer Site may be filled by this Product(\_Root).

Model	Product_Root Model
-------	--------------------

The fields 'customer' and 'customers' both specify the customer Sites that may be filled by this Product(\_Root). The 'customers' field is a Site\_Group that can contain a List of Sites. It is the general mechanism for specifying allowed customer Sites. For convenience, the field 'customer' can be used for specifying a single Site. It can be used without needing to define a Site\_Group, and thus is simpler to use. These two fields are additive: effectively the 'customer' is appended to the List in 'customers'. Both can be specified; or either can be left [unspecified]; or both can be left [unspecified], which allows any customer Site.

Forecasts for this Product(\_Root) should include only sales to Sites in 'customers' or 'customer'.  
 Default: [unspecified]

**customers** - - *a Site\_Group field of model Product\_Root*  
 Item\_Requests from the Sites in this Site\_Group may be filled by this Product(\_Root).

The fields 'customer' and 'customers' both specify the customer Sites that may be filled by this Product(\_Root). The 'customers' field is a Site\_Group that can contain a List of Sites. It is the general mechanism for specifying allowed customer Sites. For convenience, the field 'customer' can be used for specifying a single Site. It can be used without needing to define a Site\_Group, and thus is simpler to use. These two fields are additive: effectively the 'customer' is appended to the List in 'customers'. Both can be specified; or either can be left [unspecified]; or both can be left [unspecified], which allows any customer Site.

Forecasts for this Product(\_Root) should include only sales to Sites in 'customers' or 'customer'.  
 Default: [unspecified]

**forecast\_policy** - - *an extension selector of model Product\_Root*  
 Defines what forecast information will be kept in the Seller\_Plan's Forecast\_Entry's, how forecast Requests for this Product\_Root should be computed from the 'committed' Forecast such that they are representative of the expected demand, and how those forecast Requests should be adjusted as actual Requests arrive (or fail to arrive).

For example, it can define how to handle forecast error and timing variance, service level and safety level adjustments, end-of-month skewed forecast Requests, or how to expire or adjust the forecast as time passes and actual orders are received (or not). It can place orders for a Configuration that is representative of all the 'Items' included, or it can use a representative mix of the 'Items'. It may even define that forecasted mix information should be kept in each Forecast\_Entry and used to compute the representative mix.

Note that if the forecast\_policy creates forecast\_requests based on a representative item, which may be specified in a 'representative\_configuration', then that representative item MUST be specified in order for forecast\_requests to be generated properly. Failure to specify the 'representative\_configuration's item will result in forecast\_requests for the [unspecified] item of the [unspecified] site. (The only exception to this rule is the case in which the Product\_Root has only one Product\_Item. In this case, it is assumed that the single Product\_Item is intended to be the 'representative\_configuration.item' too.)

A forecast\_policy works within the forecast\_horizon of the 'top\_seller'. That is, it is independent of the granularity of member seller forecast\_horizons.

Default: SINGLE\_REQUEST

Extensions:

SIMPLE\_FIXED\_QUANTITY, SINGLE\_REQUEST, SIMPLE\_FIXED\_TIME, WEEKLY, DUAL\_REQUEST.

product, product (Product\_Allocation) -- a Product field of model

Product\_Root

Returns the Product corresponding to this Product\_Root in the specified Seller. If no Seller is specified, then this Seller is used, and this Seller's Product corresponding to this Product\_Root is returned.

Properties: Export-Only Field

unit -- a Unit field of model Product\_Root

The 'unit' defines the Quantity of one unit of this Product(\_Root), whether this Product(\_Root) is discrete (can only exist in whole units), and what the 'preferred\_measure' for this Product(\_Root) is. It can also 'convert' between different units of Measure and units of this Product(\_Root).

For example, one unit of Product X may be "3 kg", "400 ml", and "\$12". It may be 'discrete', meaning that a need for "5 kg" will result in "6 kg" (2 units) being built. The 'preferred\_measure' may be "kg" which will cause most fields that display a Quantity of this Product to display in "kg" by default.

Note that this field is not settable -- you cannot assign it a new Unit. Rather, you can get the Unit of this Product\_Root and set that Unit's fields.

Properties: Export-Only Field

owner -- a Seller field of model Product\_Root

The Seller to which this Product\_Root belongs.

Properties: Export-Only Field

5.1.2.2.1 Product\_Supplier Model

Product\_Supplier -- a submodel of model Product\_Root

The Items that are sold as this Product(\_Root), organized by supplying Site. Only Item\_Requests for these Items may be filled by this Product(\_Root). Forecasts for this Product(\_Root) only include sales of these Items.

The Product\_Supplier model has these submodels :

Product\_Item.

The Product\_Supplier model has fields that references these models :  
Site, Product\_Item, Product\_Root.

These models have a field that is a Product\_Supplier model :  
Product\_Root, Product\_Item.

The key field for this model is supplier

supplier -- a Site field of model Product\_Supplier

A Site that supplies at least one of the Items covered by this Product(\_Root).  
Default: None -- this is a key field

items -- a list of Product\_Item submodels of model Product\_Supplier  
The Items of the 'supplier' Site that are sold as the 'owner' Product(\_Root).

owner -- a Product\_Root field of model Product\_Supplier

Properties: Export-Only Field

<i>Models</i>	<i>Product_Item Model</i>
---------------	---------------------------

### 5.1.2.2.1.1 Product\_Item Model

Product\_Item -- *a submodel of model Product\_Supplier*

The Items of the 'supplier' Site that are sold as the 'owner' Product(\_Root).

The Product\_Item model has fields that references these models :  
Item, Product\_Supplier.

These models have a field that is a Product\_Item model :  
Product\_Supplier.

The key field for this model is item

item -- *a Item field of model Product\_Item*

A Request for this Item can be filled as this Product(\_Root).

Default: None -- this is a key field

owner -- *a Product\_Supplier field of model Product\_Item*

Properties: Export-Only Field

<i>Models</i>	<i>Product Model</i>
---------------	----------------------

### 5.1.2.3 Product Model

Product -- *a submodel of model Seller*

A Product defines one or more Items that are available to a set of customers with a certain delivery lead time to certain delivery territories, at a certain price. Any of those may be unlimited (any delivery lead time; any delivery address; etc.). Each Product is independently forecasted, allocated, and priced for the purposes of quoting/promising.

The primary purpose of forecasting and forecast management is to estimate the sales potentials for each of these Products, independently. The primary purpose of master planning is to determine how capacity and materials will be allocated in order to best meet the forecasts for these Products. Master planning determines how much of each Product will be available and when. Order promising is then performed in terms of those allocations to these Products. In this sense, the Products define the granularity of the Seller's master plan.

Products also define divisions for pricing, which is inherently tied to order promising (if the customer is willing to pay more, they will often find more available; similarly, if the customer is willing to wait longer, they can often get a lower price). Note, however, that the pricing rules are defined by a separate model: Seller\_Product\_Policies. This is because different Sellers may need to set pricing independently; but they should not be redefining the Product model. So, the pricing fields have been separated out. Thus, the Products essentially define the granularity of the Seller's price list, though the Seller's price list itself is defined by the Seller\_Product\_Policies models.

Note that a Customer may request a specific Item on a specific Date. That Request may be satisfied by any one of a number of Products. Each Product will have independent availability and pricing. The list of all the options for meeting the customer's Request can be presented, and the customer can choose the most satisfactory among them, based upon price and timing.

From a supply point of view, the availability of a Product is estimated based upon building a representative Configuration of an Item and delivering it to a representative Customer address. The less accurately these represent all the Items and delivery addresses included, the less accurate the availability information will be. Therefore, Promises can be padded with extra lead time to compensate for possible misrepresentation. In many cases, Products can be defined at a low enough level that availability is accurately planned and no safety padding is necessary.



Models	Product Model
--------	---------------

Products cannot be introduced directly. Instead, a Product\_Root must be added. There is one Product in a Seller for each Product\_Root defined by that Seller or any of its organization's. Certain Product characteristics can only be defined once for the whole Product-Seller tree -- those characteristics are defined by the fields of Product\_Root and are only available for viewing through the Product.

In each Seller, there is a special uneditable Product named [unspecified]. It has the default values for all fields. Since its Item is [unspecified], this Product is not real and cannot be actually requested, promised, or planned.

The model has selectors:  
forecast\_policy, expiration\_policy, allocation\_policy, availability\_policy, price\_policy.

The Product model has these submodels :  
Generic\_Product, Alternate\_Product.

The Product model has fields that references these models :  
Product\_Root, Site, Site\_Group, Generic\_Product, Alternate\_Product, Seller.

These models have a field that is a Product model :  
Seller, Product\_Root, Generic\_Product, Alternate\_Product, Sub\_Product, INDIVIDUAL, Product\_Available\_To\_Promise, Product\_Allocation.

The key field for this model is name  
This model may be extended with user-defined fields.

product\_root - - a Product\_Root field of model Product

The Product\_Root that defines the existence of this Product. Many of this Product's fields are defined by this Product\_Root. For example, 'name', 'description', 'items', 'min\_quantity', and 'min\_delivery\_lead\_time' cannot be set in this Product -- they must be set through this Product\_Root.

Properties: Export-Only Field

name - - a Symbol field of model Product

The name of this Product, which is defined by the Product\_Root. This is essentially a shortcut to 'product\_root.name'.

Models	Product Model
--------	---------------

Depending upon the environment, this is typically the name of the Item (Item family, or pseudo item), name of the Buffer (Buffer family, or SKU), or the name of the Operation. However, it is also often none of those. It simply must be unique among the Products of the 'owner' Seller, but otherwise can be whatever is meaningful.

Default: None -- this is a key field

Properties: Export-Only Field

rank - - a Number field of model Product  
Rank of this product. This is used to compute the planning priority of an item\_rap in active\_strategy.

Default: 0

description - - a String field of model Product

A description of this Product, as defined by the 'product\_root'. This is a shortcut to 'product\_root.description'.

Properties: Export-Only Field

items - - a List[Item] field of model Product

The End Items that are sold as this Product. These end items could potentially come from different sites. This List cannot be edited directly; rather, you must edit the 'Items' List under the appropriate Site in the 'suppliers' List of the 'product\_root'.

Properties: Export-Only Field

min\_quantity - - a Quantity field of model Product

An Item\_Request with a quantity less than this cannot be filled by this Product. Forecasts for this Product only include sales with this and larger Quantity. This Quantity is converted to the 'unit' of this Product's 'product\_root'.

This field cannot be edited directly; the 'min\_quantity' field of the 'product\_root' defines this value.

Properties: Export-Only Field

min\_delivery\_lead\_time - - a Time field of model Product

An Item\_Request with an delivery lead time less than this cannot be filled by this Product. Forecasts for this Product only include sales with this and larger delivery lead time. Delivery lead time is the Time from when the Promise must be accepted ('accept\_by') to when the delivery must be made ('due.end').

This field cannot be edited directly; the 'min\_delivery\_lead\_time' field of the 'product\_root' defines this value.

Properties: Export-Only Field

**delivery\_area** - - *a Symbol field of model Product*  
Defines the delivery area that this Product covers. Forecasts for this Product only include sales within this delivery area.

This field cannot be edited directly; the 'delivery\_area' field of the 'product\_root' defines this value.  
Properties: Export-Only Field

**customer** - - *a Site field of model Product*  
Item\_Requests from this customer Site may be filled by this Product.

The fields 'customer' and 'customers' both specify the customer Sites that may be filled by this Product. The 'customers' field is a Site\_Group that can contain a List of Sites. It is the general mechanism for specifying allowed customer Sites. For convenience, the field 'customer' can be used for specifying a single Site. It can be used without needing to define a Site\_Group, and thus is simpler to use. These two fields are additive: effectively the 'customer' is appended to the List in 'customers'. Both can be specified; or either can be left [unspecified]; or both can be left [unspecified], which allows any customer Site.

Forecasts for this Product should include only sales to Sites in 'customers' or 'customer'.

This field cannot be edited directly; the 'min\_delivery\_lead\_time' field of the 'product\_root' defines this value.  
Properties: Export-Only Field

**customers** - - *a Site\_Group field of model Product*  
Item\_Requests from the Sites in this Site\_Group may be filled by this Product.

The fields 'customer' and 'customers' both specify the customer Sites that may be filled by this Product. The 'customers' field is a Site\_Group that can contain a List of Sites. It is the general mechanism for specifying allowed customer Sites. For convenience, the field 'customer' can be used for specifying a single Site. It can be used without needing to define a Site\_Group, and thus is simpler to use. These two fields are additive: effectively the 'customer' is appended to the List in 'customers'. Both can be specified; or either can be left [unspecified]; or both can be left [unspecified], which allows any customer Site.

Forecasts for this Product should include only sales to Sites in 'customers' or 'customer'.

This field cannot be edited directly; the 'min\_delivery\_lead\_time' field of the 'product\_root' defines this value.  
Properties: Export-Only Field

**forecast\_policy** - - *an extension selector of model Product*  
Defines what forecast information will be kept in the Seller\_Plan's Forecast\_Entry's, how forecast Requests for this Product should be computed from the 'committed' forecast such that they are representative of the expected demand, and how those forecast Requests should be adjusted as actual Requests arrive (or fail to arrive).

For example, it can define how to handle forecast error and timing variance, service level and safety level adjustments, end-of-month skewed forecast Requests, or how to expire or adjust the forecast as time passes and actual orders are received (or not). It can place orders for a Configuration that is representative of all the 'items' included, or it can use a representative mix of the 'items'. It may even define that forecasted mix information should be kept in each Forecast\_Entry and used to compute the representative mix.

This field cannot be edited directly; the 'min\_delivery\_lead\_time' field of the 'product\_root' defines this value.

Properties: Export-Only Field  
Extensions:  
**SIMPLE\_FIXED\_QUANTITY, SINGLE\_REQUEST, SIMPLE\_FIXED\_TIME, WEEKLY, DUAL\_REQUEST.**

**active** - - *a Logical field of model Product*  
A Product is 'active' if any of it's fields has been set to other than the default value. The conversion to "active" can occur by setting a field for a Product, by activating any of a Product's specific Products (see Product\_specific\_products) or by creating an active Forecast for the Product.

The default for each of the Product model fields which return List(Problem) is to return the complete set of Products (both 'inactive' and 'active'). Each such field will also have an 'active' counterpart. For example, Seller\_products() returns all Products of the Sellers, but Seller\_active\_products() returns only active products of the Seller.

The 'active' vs 'inactive' distinction is useful in situations where the number of possible Seller/Product\_Root combinations is much larger than the number of Seller/Product\_Root combinations actively being forecasted.

The 'active' field is not settable.

Models	Product Model
--------	---------------

Properties: Export-Only Field

**expiration\_policy** - - *an extension selector of model Product*  
 Defines how the forecast for this Seller expires as time passes if the actual demand is less than expected.

The default **AT\_END** expiration\_policy, for example, does not expire any of the forecast until the end of the forecast period. So, if 1000 units are forecasted, but only 10 units have actually been requested through the 26th day of the month period, the other 990 units will continue to be forecasted to be sold in the last few days.

Default: **AT\_END**

Extensions:  
**AT\_END.**

**allocation\_policy** - - *an extension selector of model Product*  
 Defines how allocations of this Product to the 'owner' Seller are allocated to the Seller's 'members' or customers. In the absence of actual orders, this policy simply allocates to the 'members'. With actual orders present, the supplying Sites will have made allocations directly to those actual orders. Thus, the **Allocation\_Policy** can only report Problems when the allocations are inconsistent with its rules. Those Problems will need to be addressed along with all of the other Problems at each Site. The default **allocation\_policy** is **PER\_COMMITTED**. Setting the allocation policy on a Product that a Seller sells, does not recursively set the **allocation\_policy** on the products that the sub sellers sell.

For example, the **PER\_COMMITTED** policy distributes the allocations proportional to the quantity 'committed' to by that Seller. The **FIXED\_SPLIT** policy distributes according a fixed percentage breakdown among the members. When implemented, **CALENDAR\_SPLIT** will be similar, but the splits may vary over time. The **FCFS** policy retains the whole allocation at this Seller, forcing the 'members' to consume from this Seller first-come first-served (**FCFS**).

Default: **PER\_COMMITTED**

Extensions:  
**FCFS, PER\_ALLOCATED, PER\_COMMITTED, MEMBER\_RANK, FIXED\_SPLIT.**

**auto\_allocate** - - *a Logical field of model Product*  
 Specifies whether or not allocation to 'members' Sellers occurs automatically. When set to **TRUE**, changes to **Forecast\_Entry** 'allocated' quantities are immediately allocated to 'members' Sellers using this Seller's 'allocation\_policy'. When set to **FALSE**, changes to **Forecast\_Entry** 'allocated' quantities are not automatically allocated to 'members' Sellers.

Models	Product Model
--------	---------------

Default: **TRUE**

**always\_override\_members\_forecasted** - - *a Logical field of model Product*  
 Specifies whether or not forecasted quantities of member sellers are aggregated up to the seller organization. If this is "true", then the member seller quantities are not aggregated.

Default: **false**

**always\_override\_members\_committed** - - *a Logical field of model Product*  
 Specifies whether or not committed quantities of member sellers are aggregated up to the seller organization. If this is "true", then the member seller quantities are not aggregated.

Default: **false**

**availability\_policy** - - *an extension selector of model Product*  
 A new extension, called 'availability', will be added. The the following 2 extensions will be supported (initially): **SLIDING** and **HORIZON**. Specifies the method for computing product availability, (i.e. cumulative ATP) for this Product. The default availability policy, **SLIDING**, uses only the 'product.consume\_earlier' filed in computing cumulative ATP. Other availability\_policies extend the 'consume\_earlier' capability by adding concepts such as fixed bounds which limit product availability beyond fixed bounds. These bounding policies may be used to model such realities as fiscal quarters.

Default: **SLIDING**

Extensions:  
**SLIDING, HORIZON.**

**price\_policy** - - *an extension selector of model Product*  
 Specifies how the price of an Item\_Promise is computed. The Item, customer, delivery address, quantity, and delivery lead time are all typical factors used to compute the price. Some price\_policy's use Expressions which can compute from arbitrary user-defined fields on the Products, Requests, or Promises.

Default: **FIXED**

Extensions:  
**NONE, FIXED.**

consume\_earlier - - a Time field of model Product

The amount of time into the past (i.e. earlier) that is considered when calculating the ATP for a given actual promise. This time is measured from the ending date of each Forecast\_Entry (i.e. Forecast\_entry.delivery\_dates.end). The traditional definition of ATP would set this at 00, meaning that an actual promise can consume from any earlier Forecast\_Entry. However, it is not always reasonable to assume that if you can build it early, you can deliver it later.

Having the capacity to build it early does not imply the ability to build it later, either. If earlier delivery is not acceptable, that means building it early and storing it. In this case, consuming earlier ATP can result in increased storage costs. If storing the Items for excessively long periods is not acceptable (e.g., perishable Items), consuming earlier ATP can result in exceeding shelf life limits. Or if the delivery\_operation is critical, then it may not be possible to build it early, store it, and then deliver it later.

Note that Requests can give a range of delivery Dates. Anything available during that range can be consumed, plus anything available up to 'consume\_earlier' than the earliest Date in the range.

For any bucket that is completely enclosed by the Date\_Range, from the ending date of the bucket minus 'consume\_earlier' to the latest requested delivery Date, the whole bucket can be consumed. For example, if we have weekly buckets beginning at 5/1 and 5/8, and a delivery is requested between 5/12 and 5/14, and 'consume\_earlier' is set to "7 days", then the full available to promise from both the 5/1 bucket and the 5/8 bucket may be consumed. However, if 'consume\_earlier' is set to "4 days", then none of the 5/1 bucket may be consumed. Consider that months can be 28, 29, 30, and 31 days long. So, in some cases consume\_earlier of 61 days will result in 2 forecast\_entry buckets (for months of 28, 29, and 30 days) and only 1 bucket in other cases (for months of 31 days).

What this means is that certain values of consume\_earlier should not be used when using a Monthly bucket\_spec. That is, values between 56-61, 84-92, 112-124, etc. are problematic since the lower bound is evenly divisible by 28 and the upper bound is 1 shy of being evenly divisible by 31. So, when using a Monthly bucket\_spec, consume\_earlier values should therefore be 32, 63, 94, 125, etc. for 1, 2, 3, 4, etc. months of "consume\_earlier" ATP quotes.  
Default: 00

time\_pad - - a Time field of model Product

Additional Time by which the 'due' Dates quoted to customers are padded to compensate for the mis-estimation due to a poorly representative representative\_configuration. Each Seller can override this with its own Product model.  
Default: 0

quantity\_pad - - a Quantity field of model Product

Additional Quantity of this Product that is consumed for each consuming actual Request. This is used to protect against mis-estimation due to poorly representative representative\_configuration. Each Seller can override this with its own Product model. This Quantity is converted to the unit of this Product's 'product\_root'.  
Default: 0

quote\_end\_of\_bucket - - a Logical field of model Product

If this field is TRUE, the first ATP quote for this Product will have as its 'dates' the Forecast\_Entry.delivery\_date.end in which the ATP exists. If this field is FALSE, the first ATP quote for this Product will have as its 'dates' the 'request.due' date if any is available.  
Default: FALSE

quote\_multiple - - a Quantity field of model Product

ATP quotes for this Product are rounded according to this quantity. The direction of the rounding is determined by the 'quote\_larger\_multiple' field. Additionally, if the 'allocate\_quote\_multiple' field is TRUE, then allocations for this Product will also be rounded according to this quantity. This Quantity is converted to the unit of this Product's 'product\_root'.  
Default: 0

quote\_larger\_multiple - - a Logical field of model Product

If this field is TRUE, ATP quotes for this Product will be rounded up by the 'quote\_multiple' for this Product. If this field is FALSE, ATP quotes will be rounded down by the 'quote\_multiple' for this Product.  
Default: "true"/"yes"

allocate\_quote\_multiple - - a Logical field of model Product

If this field is TRUE, allocations for this Product will be rounded to the 'quote\_multiple' for this Product. This rounding is in addition to rounding according to the 'product\_root's unit which always takes place regardless of this flag. The direction (i.e. up or down) of the rounding is a function of the allocation\_policy. If this field is FALSE, no additional rounding will be done for allocations of this Product.  
Default: "true"/"yes"

Models	Product Model
--------	---------------

auto\_allocate\_from\_organization - - *a logical field of model Product*  
 If auto\_allocate\_from\_organization is "true" (the default), then changes in 'allocation' in a Forecast\_Entry affect the ATP of this Product at its organization. If auto\_allocate\_from\_organization is "false", then changes in 'allocation' in a Forecast\_Entry affects the ATP of its Generic\_Products.

Currently, this field also affects where checks against ATP\_Entry's are performed and the ordering of consumption as Promises are made. If "true", then ATP consumption is first from the Product at its 'organization' and the ATP Entries checked for this Product are those that are checked for this Product at its 'organization'. If "false", then ATP consumption is first from this Product's Generic\_Products and the ATP\_Entry's checked for this Product are those that are checked for this Product's Generic\_Products.  
 Default: "false"/"no"

generic\_products - - *a list of Generic\_Product submodels of model Product*  
 The 'generic\_products' are Products that represent more generic demand, demand for a class of products, or demand for products built with a certain component. It is often very difficult to forecast for specific Products, whereas it is much simpler and accurate to forecast for more generic Products. For example, it may be very difficult to forecast how many of each model of PC will sell; whereas, it may be much easier to forecast the number of 150MHz processors, the number of disk drives, and the number of keyboards.

Given that, it is preferable to setup allocations and ATP for those generic Products and allow all the different specific Products to consume ATP as needed from those generic Products. That prevents any need to prematurely allocate to specific Products, but still allows you to forecast and allocate some level of certain specific Products.

This is often used to implement a more sophisticated form of two-level (or N-level) master planning.

specific\_products - - *a List(Product) field of model Product*  
 The Products which name this Product as one of its 'generic\_products'. These Products are directly 'allocated', but that Quantity is included in this Product's 'allocated' Quantity. This Product is managed as a more generic Product that includes the demand for these 'specific\_products'. The 'specific\_products' are subsets of this generic Product.  
 Properties:   Export-Only Field

Models	Product Model
--------	---------------

active\_specific\_products - - *a List(Product) field of model Product*  
 Returns active Products on the specific\_products list. See Product.active\_specific\_products && Product.active.  
 Properties:   Export-Only Field

alternate\_products - - *a list of Alternate\_Product submodels of model Product*  
 The 'alternate\_products' could be used as the alternates of this product. This is different from the various products of an 'item' in the sense that the alternate products could be represented as an alternate for a generic product or as an alternate for a SKU that may have some generic products as well.

primary\_products - - *a List(Product) field of model Product*  
 The Products which name this Product as one of its 'alternate\_products'. These Products are directly 'allocated'.  
 Properties:   Export-Only Field

active\_primary\_products - - *a List(Product) field of model Product*  
 Returns active Products on the primary\_products list. See Product.primary\_products && Product.active.  
 Properties:   Export-Only Field

groups - - *a List(Product\_Group) field of model Product*  
 The Product\_Groups to which this Product belongs. No Product may appear in more than one Product\_Group of a single Product\_Group hierarchy. However, a Product may appear in any number of separate Product\_Group hierarchies.

A Product that appears in no Product\_Groups is considered to be its own product hierarchy, and will appear in the 'owner' Seller's 'top\_products'.

This field cannot be edited directly; this List is determined by which Product\_Groups specify this as a Sub\_Product.  
 Properties:   Export-Only Field

top - - *a logical field of model Product*  
 If "true", then this Product appears in no Product\_Groups -- it is the top of a Product hierarchy.  
 This field cannot be edited directly; this List is determined by which Product\_Groups specify this as a Sub\_Product.  
 Properties:   Export-Only Field

Models	Product Model
--------	---------------

owner - - *a Seller field of model Product*

Properties:    Export-Only Field

Models	Generic_Product Model
--------	-----------------------

5.1.2.3.1    Generic\_Product    Model

Generic\_Product -- *a submodel of model Product*

The 'generic\_products' are Products that represent more generic demand, demand for a class of products, or demand for products built with a certain component. It is often very difficult to forecast for specific Products, whereas it is much simpler and accurate to forecast for more generic Products. For example, it may be very difficult to forecast how many of each model of PC will sell, whereas, it may be much easier to forecast the number of 150MHz processors, the number of disk drives, and the number of keyboards.

Given that, it is preferable to setup allocations and ATP for those generic Products and allow all the different specific Products to consume ATP as needed from those generic Products. That prevents any need to prematurely allocate to specific Products, but still allows you to forecast and allocate some level of certain specific Products.

This is often used to implement a more sophisticated form of two-level (or N-level) master planning.

The Generic\_Product model has fields that references these models :  
Product.

These models have a field that is a Generic\_Product model :  
Product.

The key field for this model is product

product - - *a Product field of model Generic\_Product*  
The Product that represents the demand for a generic of this Product. This 'product's ATP can be consumed by the 'owner' Product and offered as its own (assuming all the Generic\_Products have corresponding ATP available).  
Default:    None -- this is a key field

quantity\_per - - *a Quantity field of model Generic\_Product*  
The Quantity of 'product' that must be consumed per unit of the 'owner' Product. This is converted to the 'product.unit'; not the 'owner.unit'.  
Default:    1

owner - - *a Product field of model Generic\_Product*

Properties:    Export-Only Field

Models	Alternate_Product Model
--------	-------------------------

### 5.1.2.3.2 Alternate\_Product Model

#### Alternate\_Product -- a submodel of model Product

The 'alternate\_products' could be used as the alternates of this product. This is different from the various products of an 'item' in the sense that the alternate products could be represented as an alternate for a generic product or as an alternate for a SKU that may have some generic products as well.

The Alternate\_Product model has fields that references these models :

Product.

These models have a field that is a Alternate\_Product model :

Product.

The key field for this model is product

product -- a *Product field of model Alternate\_Product*  
The Product that is an alternate for this Product.

Default: None -- this is a key field

quantity\_per -- a *Quantity field of model Alternate\_Product*

The Quantity of 'product' that must be consumed per unit of the 'owner' Product. This is converted to the 'product.unit'; not the 'owner.unit'.

Default: 1

rank -- a *Number field of model Alternate\_Product*

Rank of this alternate product. This may be used in quoting and consumption to determine how to consume ATP from a group of alternates.

Default: 0

owner -- a *Product field of model Alternate\_Product*

Properties: Export-Only Field

Models	allocation_policy submodels of model Product
--------	--

### 5.1.2.3.3 allocation\_policy submodels of model Product

#### 5.1.2.3.4 Product\_Allocation Model

#### Product\_Allocation -- a submodel of model Product

The 'allocations' or fixed percentages which are used by this product's owner when splitting its allocated amount up among its members. Any entry of 'allocations' whose split is set to zero is automatically deleted.

The Product\_Allocation model has fields that references these models :

Seller, Product.

These models have a field that is a Product\_Allocation model :

FIXED\_SPLIT.

The key field for this model is member

member -- a *Seller field of model Product\_Allocation*

A member of this Product's owner for which a fixed allocation percentage is defined.

Default: None -- this is a key field

split -- a *Percentage field of model Product\_Allocation*

The fixed percentage split of any amount allocated to this Product's owner which should be allocated to this 'member'. The 'split' may not be less than zero.

Note that the 'split's of the 'allocations' of a Product need not add up to "100%". If there are four 'allocations' that each have a 'split' of "100%", giving a total of "400%", the split will be even -- the same as if each had a split of "25%" for a total of "100%". Thus, the default, "100%", will result in an even split.

Default: 100%

owner -- a *Product field of model Product\_Allocation*

Properties: Export-Only Field

Models	Product_Group Model
--------	---------------------

### 5.1.2.4 Product\_Group Model

#### Product\_Group -- a submodel of model Seller

A Product\_Group models a hierarchical grouping or classification of Products. A Product can only appear directly in only one Product\_Group of a hierarchy. Similarly, a Product\_Group can only appear directly in one place in the hierarchy. However, a Product can appear in any number of separate Product\_Group hierarchies.

An implication of this is that all the Product\_Groups cannot be assembled into the one [unspecified] Product\_Group -- the hierarchies must remain separate, otherwise the totals for one Product would be counted dozens of times or the totals would not add in an understandable way.

For example, milk Products may be grouped by fat content: "skim", "1%", "2%", and "whole". The same milk Products may also be grouped by size: pint, quart, half-gallon, and gallon. Further, the same milk Products may be grouped by brand: "Econo-Cow" and "Pure-White". A particular Product would be in one fat content group, one size group, and one brand group.

Product\_Groups are organized into hierarchies. In the above example, the four fat groups might be in the "Milk by Fat" Product\_Group, the four sizes might be in the "Milk By Size" group, and the two brands might be in the "Milk By Brands" group. Further, if there are non-milk Products, those groups may be included into other Product\_Groups. For instance, "Milk By Size" may be in the "By Drink By Size" Product\_Group, which also includes "Orange\_Juice\_By\_Size" and others.

Note that grouping "Milk by Fat", "Milk by Size", and "Milk by Brands" into a "Milk" group would result in counting each Milk product as many as three times. Each "Milk by \*" group gives the summary of all Milk, but then breaks it out different ways.

All of the above milk Product\_Groups are related to Item characteristics. You can also form Product\_Groups based on other properties of the Products. For instance, your Products may be divided based upon groups of customers (Site\_Groups), or "market channels", or "industry segments". For example, pints of milk sold to grocery stores may be a separate Product from pints of milk sold to restaurants. A Product\_Group may be formed based on that division: "Milk By Channel".

Another common division will be by delivery lead time. Often the same Item will be made available as one Product with a 2-week delivery lead time and another with a 6-week delivery lead time, where the latter has a significantly lower price. When Products are so divided, they will almost always be grouped together for forecast purposes.

Models	Product_Group Model
--------	---------------------

This organization provides a great deal of flexibility in organizing and displaying information about the Products in the interactive Reports. Since Forecast accuracy improves with aggregation, being able to aggregate in many different dimensions is a key capability in forecast management. That is the purpose of Product\_Group.

The Product\_Group model has these submodels :

Sub\_Product\_Group, Sub\_Product.

The Product\_Group model has fields that references these models :  
Product\_Group, Sub\_Product\_Group, Unit, Seller.

These models have a field that is a Product\_Group model :  
Seller, Product\_Group, Sub\_Product\_Group, Sub\_Product, GROUP.

The key field for this model is name  
This model may be extended with user-defined fields.

name -- a Symbol field of model Product\_Group  
The name of this Product\_Group. This name must be unique among Product\_Groups of the owner Seller.

Default: None -- this is a key field

description -- a String field of model Product\_Group

A description of this Product\_Group.

Default: none

top -- a Logical field of model Product\_Group

If "true", then this Product\_Group appears in no other Product\_Groups -- it is the top of a Product\_Group hierarchy.

Properties: Export-Only Field

root -- a Product\_Group field of model Product\_Group

The root of this Product\_Group-Seller tree; the corresponding Product\_Group of the highest level Seller that defines a Product\_Group with the same name.

Note that it is only possible to delete a Product\_Group or change its name if it is the root of its Product\_Group-Seller tree.

Properties: Export-Only Field



Models	Product_Group Model
--------	---------------------

use\_sid\_split - - a logical field of model Product\_Group  
 If "true", then any addition or reduction to an forecast entry of a product\_group should be distributed among its 'sub\_groups / sub\_products' using the 'sid\_split' specified in the Product\_Group.

If "false", then setting an forecast entry value for this product\_group should distribute it to the 'sub\_groups / sub\_products' such that the Percentage split among the 'sub\_groups / sub\_products' remain the same.

For example, consider a Product\_Group that contains A, B, and C, and specifies the 'sid\_split's to be "60%", "30%", and "10%", respectively. Further, assume that for one Date\_Range the current Forecasts are "20", "16", and "4", respectively, for a group total of "40". The group total is then set to "50". If use\_sid\_split is "false", then the current split will be followed, resulting in "25", "20", and "5", respectively. If use\_sid\_split is "true", then the additional "10" (50 - 40) will be distributed with the 'sid\_split's, adding "6", "3", and "1", resulting in "26", "19", and "5", respectively.  
 Default: false

sub\_groups - - a list of Sub\_Product\_Group submodels of model Product\_Group  
 The Product\_Groups that are direct members of this Product\_Group. Note that no Product\_Group descendant can contain this Product\_Group or contain common Products.

sub\_products - - a list of Sub\_Product submodels of model Product\_Group  
 The Products that are direct members of this Product\_Group. This does not include the Products in this Product\_Group's 'sub\_products'. The 'all\_products' field provides a complete list.

all\_products, all\_products (List(Product)) - - a List(Product) field of model Product\_Group  
 All of the Products that are in this Product\_Group, both directly and indirectly. This list includes the Products in 'sub\_products' plus 'all\_products' of each of the Product\_Groups in 'sub\_groups'.

If passed List(Product), it returns only those Products that are in the passed List and in this Product\_Group. In other words, it returns the intersection of the Products in this Product\_Group and the argument List.  
 Properties: Export-Only Field

Models	Product_Group Model
--------	---------------------

all\_products\_and\_specifics - - a List(Product) field of model Product\_Group  
 This list includes the Products in 'sub\_products' plus 'all\_products' of each of the Product\_Groups in 'sub\_groups' plus all 'generic\_products' and 'specific\_products' in the case of 'sub\_products' having related products.  
 Properties: Export-Only Field

unit - - a Unit field of model Product\_Group  
 The unit defines the Quantity of one unit of this Product\_Group, whether this Product\_Group is 'discrete' (can only exist in whole units), and what the 'preferred\_measure' for this Product\_Group is. It can also 'convert' between different units of Measure and units of this Product\_Group.

For example, one unit of Product\_Group X may be "3 kg", "400 ml", and "\$12". It may be 'discrete', meaning that a need for "5 kg" will result in "6 kg" (2 units) being built. The 'preferred\_measure' may be "kg" which will cause most fields that display a Quantity of this Product\_Group to display in "kg" by default.

Note that this field is not settable -- you cannot assign it a new Unit. Rather, you can get the Unit of this Product\_Group and set that Unit's fields.  
 Properties: Export-Only Field

owner - - a Seller field of model Product\_Group  
 The Seller to which this Product\_Group belongs.  
 Properties: Export-Only Field

Models	Sub_Product_Group Model
--------	-------------------------

### 5.1.2.4.1 Sub\_Product\_Group Model

#### Sub\_Product\_Group -- a *submodel* of model Product\_Group

The Product\_Groups that are direct members of this Product\_Group. Note that no Product\_Group descendant can contain this Product\_Group or contain common Products.

The Sub\_Product\_Group model has fields that references these models :  
Product\_Group, Sub\_Product\_Group.

These models have a field that is a Sub\_Product\_Group model :  
Product\_Group, Sub\_Product\_Group.

The key field for this model is product\_group

**product\_group** -- a *Product\_Group field of model* Sub\_Product\_Group  
A Product\_Group that is a direct member of the 'owner' Product\_Group.  
Default: None -- this is a key field

**quantity\_per** -- a *Quantity field of model* Sub\_Product\_Group  
The Quantity of 'product\_group' per unit of the 'owner' Product\_Group. For example, one unit of the 'owner' Product\_Group may be "10 gallon", "20 kg", "\$4", or "3" (three units) of this sub\_group. The equivalence among sub\_groups must be defined in order for aggregate forecasting to be meaningful. Very often, however, the base units of all Product\_Groups are the same, and thus the default of "1" (one unit to one unit) is correct. This Quantity is converted to the 'unit' of the 'product\_group' (the "sub" group), not of the 'owner' Product\_Group.  
Default: 1

**std\_split** -- a *Percentage field of model* Sub\_Product\_Group  
The typical percentage of the 'owner' Product\_Group's Forecast that should be allocated to this 'product\_group'. This can be used (but need not be) by GROUP Forecasts in allocating to its 'sub\_forecasts'.

Note that the 'std\_split's of the 'sub\_groups' of a Product\_Group need not add up to "100%". If there are four 'sub\_groups' that each have a 'std\_split' of "100%", giving a total of "400%", the split will be even -- the same as if each had a split of "25%" for a total of "100%". Thus, the default, "100%", will result in an even split.  
Default: 100%

Models	Sub_Product_Group Model
--------	-------------------------

**root** -- a *Sub\_Product\_Group field of model* Sub\_Product\_Group  
The root of this Sub\_Product\_Group-Seller tree; the corresponding Sub\_Product\_Group of the highest level Seller that defines this 'product\_group' to be a Sub\_Product\_Group of the corresponding 'owner' Product\_Group.

Note that it is only possible to delete a Sub\_Product\_Group or change its 'product\_group' if it is the root of its Sub\_Product\_Group-Seller tree.  
Properties: Export-Only Field

**owner** -- a *Product\_Group field of model* Sub\_Product\_Group

Properties: Export-Only Field

Models	Sub_Product Model
--------	-------------------

### 5.1.2.4.2 Sub\_Product Model

#### Sub\_Product -- a submodel of model Product\_Group

The Products that are direct members of this Product\_Group. This does not include the Products in this Product\_Group's 'sub\_products'. The 'all\_products' field provides a complete list.

The Sub\_Product model has fields that references these models :  
Product, Sub\_Product, Product\_Group.

These models have a field that is a Sub\_Product model :  
Product\_Group, Sub\_Product.

The key field for this model is product

product - - a Product field of model Sub\_Product  
A Product that is a direct member of the 'owner' Product\_Group.  
Default: None -- this is a key field

quantity\_per - - a Quantity field of model Sub\_Product  
The Quantity of product per unit of this Product\_Group. For example, one unit of the 'owner' Product\_Group may be "10 gallon", "20 kg", "\$4", or "3" (three units) of this product. The equivalence among sub\_products and the 'owner' Product\_Group must be defined in order for aggregate forecasting to be meaningful. Very often, however, the base units of all Product\_Groups are the same, and thus the default of "1" (one unit to one unit) is correct. This Quantity is converted to the 'unit' of the product (the "sub product), not of the 'owner' Product\_Group.  
Default: 1

sid\_split - - a Percentage field of model Sub\_Product  
The typical percentage of the 'owner' Product\_Group's Forecast that should be allocated to this product. This can be used (but need not be) by GROUP Forecasts in allocating to its 'sub\_forecasts'. Note that if the splits add up to more than 100%, they will be normalized to 100% when used to split the forecast.  
Default: 1

root - - a Sub\_Product field of model Sub\_Product  
The root of this Sub\_Product-Seller tree; the corresponding Sub\_Product of the highest level Seller that defines this product to be a Sub\_Product of the corresponding 'owner' Product\_Group.

Models	Sub_Product Model
--------	-------------------

Note that it is only possible to delete a Sub\_Product or change its 'product' if it is the root of its Sub\_Product-Seller tree.  
Properties: Export-Only Field

owner - - a Product\_Group field of model Sub\_Product  
Properties: Export-Only Field

Models	Plan Model
--------	------------

### 5.1.3 Plan Model

Plan -- *a submodel of model Supply\_Chain*

The Plan models what is happening and what is planned to be done in a Supply\_Chain. The Supply\_Chain defines what is possible; the Plan defines the current plans for using the Site capabilities.

Plan's structure mirrors that of Supply\_Chain. As Supply\_Chain is made up of Sites, Plan is made up of Site\_Plans. As Site is made up of Operations, Resources, and Buffers, Site\_Plan is made up of Operation\_Plans, Resource\_Plans, and Buffer\_Plans. Supply\_Chain contains the "master" data; State contains the "live" data.

Plans define an Edit-Boundary: edit-security and edit-update modification for all models under Plan are controllable at the Plan level.

This split is useful for divisions to support What-If analysis. You may want to consider a particular Supply\_Chain in different states with different plans. For example, what if an oven Resource goes down? What if we receive triple the demand we have forecasted? What if we buy a new machine? What if we run these jobs earlier? What if these arrive late?

The state specification mechanisms of the Plan models are designed to support the "minimal transaction" philosophies, such as Just-In-Time (JIT). As time marches forward, we simply assume everything is going according to the Plan, unless told otherwise in the form of state "overrides".

The Plan model has these submodels :  
Site\_Plan, Seller\_Plan, Problem, Active\_Strategy.

The Plan model has fields that references these models :  
Site\_Plan, Seller\_Plan, Problem, Active\_Strategy, Resource\_Plan, Buffer\_Plan, Forecast, Supply\_Chain.

These models have a field that is a Plan model :  
Supply\_Chain, Site\_Plan, Seller\_Plan, Problem, Active\_Strategy.

The key field for this model is name  
This model may be extended with user-defined fields.

name - - *a Symbol field of model Plan*  
The name of this Plan.

Models	Plan Model
--------	------------

Default: None -- this is a key field

description - - *a String field of model Plan*  
A description of this Plan.

Default: none

current - - *a Date field of model Plan*  
The Date for which this Plan is being computed; the Date this Plan is expected to be put into place. Plans before this Date should generally not be adjusted since it will be too late by the time this Plan is put into place.

If the current planning effort is trying to prepare a Plan that will go into effect tomorrow morning, then Plan::current should be set to tomorrow morning such that the plans between 'now' and tomorrow morning are treated as if they have already occurred, and therefore are not open for replanning.

If the 'current' Date is set beyond the 'horizon', the 'horizon' will be expanded to enclose 'current'.

Note that this Date moves independent of 'now' and 'horizon'. The 'current' Date will move with the most detailed planning cycle, whereas 'horizon' will move with the least detailed. For example,

Note that there is no undo from setting 'current'. For this reason, it is a good idea to save your plan prior to setting 'current'.  
Default: 'now'

horizon - - *a Date\_Range field of model Plan*  
The planning horizon of this Plan. The Date\_Range over which this Plan spans.

The 'horizon.end', which must be after 'current', defines the amount of time the Plan extends into the future. The 'horizon.start', which must be at or before 'current', defines the amount of time the Plan extends into the past.

Note that these dates move independent of 'now' and 'current'. Generally, these Dates are moved in conjunction with a regular planning cycle. For example, the 'horizon' will be moved forward each month in conjunction with importing the new forecast information which is computed monthly. In contrast, 'current' may move forward daily our hourly during that monthly planning cycle, as detailed adjustments are made. Traditional software that does not support multiple granularities often does not need a horizon into the past since 'current' is as far back as it needs to go.

Models	Plan Model
--------	------------

Note that there is no undo from setting 'horizon'. For this reason, it is a good idea to save your plan prior to setting 'horizon'.

Default: from the start of this month until a year from then

default\_operation\_plan\_rank - - a Expression field of model Plan

An Expression that is passed an Operation\_Plan as '#' and is executed whenever a new top Operation\_Plan is created. Every top Operation\_Plan created will get its rank assigned upon creation in this manner.

Default: 0

site\_plans - - a list of Site\_Plan submodels of model Plan

The plans for each Site defined by the 'owner' Supply Chain. Each site will have a Site\_Plan per Plan in the owner Supply\_Chain. So, if the Supply\_Chain has two plans(one production plan and the other what-if plan), each site will have two Site\_Plans, one each per Plan.

top\_site\_plans - - a List(Site\_Plan) field of model Plan

The plans for each Site in the owner Supply\_Chain's list of 'top\_sites'. These Sites are the topmost organizations; they are not within another organization. This list is particularly useful as a starting point for displaying the hierarchy of Site\_Plans.

Properties: Export-Only Field

seller\_plans - - a list of Seller\_Plan submodels of model Plan

The plans for the sales personnel and sales organizations responsible for forecasting and selling this 'supply\_chain's Products. It includes those Forecasts, the Requests generated by those Forecasts, and customer Requests through those Sellers.

top\_seller\_plans - - a List(Seller\_Plan) field of model Plan

The plans for each Seller in the owner Supply\_Chain's list of 'top\_sellers'. These Sellers are the topmost organizations; they are not within another organization. This list is particularly useful as a starting point for displaying the hierarchy of Seller\_Plans.

Properties: Export-Only Field

the\_problems - - a list of Problem submodels of model Plan

The Problems that have been detected in this Plan. These may be feasibility Problems, or simply undesirable situations (such as extra cost incurred to run overtime).

problems, problems (Date\_Range) , problems (Problem\_Category) , problems (Date\_Range, Problem\_Category) - - a List(Problem) field of model Plan

The Problems detected with this Plan. If passed a Date\_Range, only the Problems whose 'dates' overlap are returned. If passed a Problem\_Category, only the Problems with that 'category' are returned.

Models	Plan Model
--------	------------

Note that you can pass in one of the special Problem\_Category's to get Problems in larger groups. For example, the OPERATION Problem\_Category will give all Problems in Operation-related Problem categories. Similarly for RESOURCE, BUFFER, and even ALL.

Properties: Export-Only Field

problem\_categories - - a List(Problem\_Category) field of model Plan

For each different 'category' of Problem in 'problems', a Problem\_Category is added to this list. It gives you the name of the 'category' (in various forms) plus a list of just the Problems of that category'. These sublists are often much easier to deal with than the full 'problems' list.

Properties: Export-Only Field

top\_active\_strategies - - a List(Active\_Strategy) field of model Plan

A list of the top level Active\_Strategies in the plan.

Properties: Export-Only Field

active\_strategies - - a list of Active\_Strategy submodels of model Plan

Each Problem\_Set defines a set of Problems to be addressed by this Active\_Strategy. The Problems are specified by category and tolerances within a certain horizon. The relative 'focus' that the Active\_Strategy will place on each Problem is also specified.

auto\_run\_strategy - - a Active\_Strategy field of model Plan

The Active\_Strategy that is specified with 'auto\_run="true"'. It is run automatically after each plan change.

Properties: Export-Only Field

background\_run\_strategy - - a Active\_Strategy field of model Plan

The Active\_Strategy that is specified with 'background\_run="true"', if there is one. It is run automatically when the planning engine is otherwise idle.

Properties: Export-Only Field

resource\_plan (Resource) - - a Resource\_Plan field of model Plan

Returns the Resource\_Plan of this Plan for the specified Resource.

Note that 'plan.resource\_plan(resource)' is equivalent to 'plan.site\_plans.find(resource.owner).resource\_plans.find(resource)'.  
Properties: Export-Only Field

Models	Plan Model
--------	------------

**buffer\_plan (Buffer)** - - *a Buffer\_Plan field of model Plan*  
Returns the Buffer\_Plan of this Plan for the specified Buffer.

Note that 'plan.buffer\_plan(buffer)' is equivalent to  
'plan.site\_plans.find(buffer.owner).buffer\_plans.find(buffer)';  
Properties: Export-Only Field

**operation\_plans (Operation)** - - *a List(Operation\_Plan) field of model Plan*  
Returns the Operation\_Plans of this Plan for the specified Operation.

Note that 'plan.operation\_plans(operation)' is equivalent to 'plan.site\_plans.find(operation.owner).operation\_plans.filter{#operation == operation}';  
Properties: Export-Only Field

**forecast (Product)** - - *a Forecast field of model Plan*  
Returns the Forecast of this Plan for the specified Product.

Note that 'plan.forecast(product)' is equivalent to 'plan.seller\_plans.find(product.owner).forecasts.find(product)';  
Properties: Export-Only Field

**run\_for\_now (Integer)** - - *a Void field of model Plan*  
Temporary run field for now.  
Properties: command=True Export-Only Field

**owner** - - *a Supply\_Chain field of model Plan*  
Properties: Export-Only Field

Models	Site_Plan Model
--------	-----------------

### 5.1.3.1 Site\_Plan Model

**Site\_Plan** - - *a submodel of model Plan*

The Site\_Plan models the plans for a Site. As Sites can be made up of Operations, Resources, and Buffers, so can Site\_Plans can be made up of Operation\_Plans, Resource\_Plans, and Buffer\_Plans.

The model has selectors:  
role.

The Site\_Plan model has fields that references these models :  
Site, Site\_Plan.

These models have a field that is a Site\_Plan model :  
Site, Plan, Operation\_Plan, Resource\_Plan, Buffer\_Plan, Site\_Plan, Operation\_State, Request, Promise, Acceptance, Delivery\_Request.

The key field for this model is site  
This model may be extended with user-defined fields.

site - - *a Site field of model Site\_Plan*  
The Site for which this plan pertains.  
Default: None - - this is a key field

description - - *a String field of model Site\_Plan*  
A description of this Site\_Plan.  
Default: none

organization\_plan - - *a Site\_Plan field of model Site\_Plan*  
The Site\_Plan for this site's organization';  
Properties: Export-Only Field

members - - *a List(Site\_Plan) field of model Site\_Plan*  
The Site\_Plans for Sites which specify this Site as their 'organization';  
Properties: Export-Only Field

role - - *an extension selector of model Site\_Plan*  
This is the same value as the 'site.role'. Here, it adds to the Site\_Plan the fields associated with the Site's role in the Supply\_Chain. For instance, CUSTOMER Sites do not have 'requests' and 'promises', but LINK and SUPPLIER Sites do.

Properties: extension\_same\_as\_model=Site Export-Only Field  
Extensions:  
LINK, SUPPLIER, CUSTOMER.

owner - - a Plan field of model Site\_Plan

Properties: Export-Only Field

5.1.3.1.1 Standard Extensions of model Site\_Plan  
5.1.3.1.1.1 role extensions of model Site\_Plan  
5.1.3.1.1.1  
LINK -- a role extension of model Site\_Plan

A LINK Site is considered to be a "link" of this supply "chain".

The LINK model has these submodels :  
Buffer\_Plan, Resource\_Plan, Operation\_Plan, Operation\_State, Request, Promise, Acceptance.

The LINK model has fields that references these models :  
Buffer\_Plan, Resource\_Plan, Operation\_Plan, Operation\_State, Request, Promise, Acceptance.

buffer\_plans - - a list of Buffer\_Plan submodels of model LINK  
The plans for each Buffer defined in the site\_plan site. Each buffer in each Site will have exactly one Buffer\_Plan per Site\_Plan.

resource\_plans - - a list of Resource\_Plan submodels of model LINK  
The plans for each Resource defined in this site\_plan's site. Each resource in each Site will have one Resource\_Plan per Site\_Plan.

operation\_plans - - a list of Operation\_Plan submodels of model LINK  
The plans for each Operation defined in this site\_plan's site. Each operation in each Site could have multiple Operation\_Plans per Site\_Plan.

operation\_states - - a list of Operation\_State submodels of model LINK  
State reports which will be assigned to Operation\_Plans.

requests - - a list of Request submodels of model LINK  
The Requests that have been made by other Sites for Items supplied by this Site. This is the raw demand on this Site.

promises - - a list of Promise submodels of model LINK  
The promises that have been made by this Site to supply Items to other Sites. This is the demand this Site has agreed to fulfill. By some definitions, this is the master production schedule.

acceptances - - a list of Acceptance submodels of model LINK  
The acceptances that have been made by the other Site in response to the promises from this site

Models	LINK Extension
--------	----------------

**plan\_to\_satisfy\_unanswered\_requests** - - *a Void field of model LINK*  
 This command creates plans to satisfy all the Requests that have been made upon this Site\_Plan that have not yet been "answered". It does this by simply calling plan\_to\_satisfy' on each of its "unanswered" requests'.

Alternatively, plan\_to\_satisfy\_all\_promises' can create plans such that all Promises would be satisfied; or plan\_to\_satisfy\_all\_requests' to create plans such that all Requests upon this Site\_Plan are satisfied.  
 Properties:    command=True    Export-Only Field

**plan\_to\_satisfy\_queued\_requests** - - *a Void field of model LINK*  
 This command creates plans to satisfy all the Requests that have been made upon this Site\_Plan that have been "queued" for later consideration. It does this by simply calling plan\_to\_satisfy' on each of its "queued" requests'.

Alternatively, plan\_to\_satisfy\_all\_promises' can create plans such that all Promises would be satisfied; or plan\_to\_satisfy\_all\_requests' to create plans such that all Requests upon this Site\_Plan are satisfied.  
 Properties:    command=True    Export-Only Field

**plan\_to\_satisfy\_all\_requests** - - *a Void field of model LINK*  
 This command creates plans to satisfy all the Requests that have been made upon this Site\_Plan. It does this by simply calling plan\_to\_satisfy' on each of its requests'. Thus, it is equivalent to requests\_for\_each(#,plan\_to\_satisfy)'.  
 Alternatively, plan\_to\_satisfy\_all\_promises' can create plans such that all Promises would be satisfied; or plan\_to\_satisfy\_unanswered\_requests' to create plans such that Requests that have not yet been "answered" are satisfied.  
 Properties:    command=True    Export-Only Field

**plan\_to\_satisfy\_all\_promises** - - *a Void field of model LINK*  
 This command creates plans to satisfy all the Promises that have been made by this Site\_Plan. It does this by simply calling plan\_to\_satisfy' on each of its promises'. Thus, it is equivalent to promises\_for\_each(#,plan\_to\_satisfy)'.  
 Alternatively, plan\_to\_satisfy\_all\_requests' can create plans such that all Requests would be satisfied; or plan\_to\_satisfy\_unanswered\_requests' to create plans such that Requests that have not yet been "answered" are satisfied.  
 Properties:    command=True    Export-Only Field

Models	LINK Extension
--------	----------------

**promise\_as\_planned** - - *a Void field of model LINK*  
 This command sets this Site\_Plan's promises' to promise what has been planned for each. It does this by simply calling promise\_as\_planned' on each of its promises'. Thus, it is equivalent to promises\_for\_each(#,promise\_as\_planned)'.  
 Properties:    command=True    Export-Only Field

**supply\_requests** - - *a List(Request) field of model LINK*  
 The Requests that have been made by this Site for Items supplied by other Sites.  
 Properties:    Export-Only Field

**supply\_promises** - - *a List(Promise) field of model LINK*  
 The Promises that have been made by other Sites to supply Items to this Site.  
 Properties:    Export-Only Field

**supply\_item\_requests**, **supply\_item\_requests** (Date\_Range) ,  
**supply\_item\_requests** (Item) , **supply\_item\_requests** (Item, Date\_Range) - - *a List(Item\_Request) field of model LINK*  
 The Requests that have been made by this Site for Items supplied by other Sites.  
 Properties:    Export-Only Field

**item\_demand** (List(Item), Date\_Range) - - *a Quantity field of model LINK*  
 Returns the sum of demand for a List of Items during a Date\_Range. The Quantity will be unitless as the sum of the number of units of each Item.  
 Properties:    Export-Only Field

**problems**, **problems** (Date\_Range) , **problems** (Problem\_Category) , **problems** (Problem\_Category, Date\_Range) - - *a List(Problem) field of model LINK*  
 The Problems detected with this Site\_Plan. If passed a Problem\_Category, only the Problems with that category' are returned. If passed a Date\_Range, only the Problems whose dates overlap are returned.

Note that you can pass in one of the special Problem\_Category's to get Problems in larger groups. For example, the OPERATION Problem\_Category will give all Problems in Operation-related Problem categories. Similarly for RESOURCE, BUFFER, and even ALL.  
 Properties:    Export-Only Field



Models	SUPPLIER Extension
--------	--------------------

problem, categories - - a List(Problem\_Category) field of model LINK  
 For each different 'category' of Problem in 'problems', a Problem\_Category is added to this list. It gives you the name of the 'category' (in various forms) plus a list of just the Problems of that 'category'. These sublists are often much easier to deal with than the full 'problems' list.

Note that this List will not include special Problem\_Categories, such as OPERATION, RESOURCE, BUFFER, or ALL.

Properties: Export-Only Field

### 5.1.3.1.1.2

SUPPLIER -- a role extension of model Site\_Plan

A SUPPLIER Site is not planned or modeled in detail; rather it represents the Items that can be procured from a supplier by LINK Sites, the Requests for those Items, and the Promises received from the supplier.

The SUPPLIER model has these submodels :  
 Request, Promise, Acceptance.

The SUPPLIER model has fields that references these models :  
 Request, Promise, Acceptance.

requests - - a list of Request submodels of model SUPPLIER  
 The Requests that have been made by other Sites for Items supplied by this Site. This is the raw demand on this Site.

promises - - a list of Promise submodels of model SUPPLIER  
 The promises that have been made by this Site to supply Items to other Sites. This is the demand this Site has agreed to fulfill.

acceptances - - a list of Acceptance submodels of model SUPPLIER  
 The acceptances that have been made by the other Site in response to the promises from this site

### 5.1.3.1.1.3

CUSTOMER -- a role extension of model Site\_Plan

A CUSTOMER Site is not planned or modeled in detail; rather, it is simply a destination for deliveries and source of Requests.

Models	role submodels of model Site_Plan
--------	-----------------------------------

### 5.1.3.1.2 role submodels of model Site\_Plan

#### 5.1.3.1.3 Buffer\_Plan Model

Buffer\_Plan -- a submodel of model Site\_Plan

A Buffer is a model of the management of an Item at a particular Location (a SKU). Most material/inventory planning functionality is handled by Buffer.

The model has selectors:  
 flow\_policy.

The Buffer\_Plan model has these submodels :  
 Lot.

The Buffer\_Plan model has fields that references these models :  
 Buffer, Lot, Operation\_Plan, Site\_Plan.

These models have a field that is a Buffer\_Plan model :  
 Plan, Buffer, LINK, Flow\_Plan, Lot, PRODUCE, CONSUME, NEGATIVE\_ON\_HAND, OVER\_FLOW\_LIMIT, NEGATIVE\_ON\_HAND\_AT\_END, LOT\_OVER\_CONSUMED, LOT\_NOT\_CONSUMED, LOT\_NOT\_PRODUCED, LOT\_OVER\_PRODUCED, LOW\_ON\_HAND, EXCESS\_ON\_HAND, EXCESS\_ON\_HAND\_AT\_END, BUFFER\_CHARGES, Sorted\_Bucket.

The key field for this model is buffer  
 This model may be extended with user-defined fields.

buffer - - a Buffer field of model Buffer\_Plan  
 The Buffer that this is planning.  
 Default: None -- this is a key field

remark - - a String field of model Buffer\_Plan  
 Comments about this specific plan.  
 Default: none

flow\_plans, flow\_plans (Date\_Range) , flow\_plans (Date\_Range, Logical) - - a List(Flow\_Plan) field of model Buffer\_Plan  
 The Flows planned to produce into or consume from this Buffer.  
 Properties: Export-Only Field

**producing\_flow\_plans, producing\_flow\_plans (Date\_Range) ,  
producing\_flow\_plans (Date\_Range, Logical) -- a List[Flow\_Plan] field of model  
Buffer\_Plan**  
The Flows planned to produce into this Buffer.  
Properties: Export-Only Field

**consuming\_flow\_plans, consuming\_flow\_plans (Date\_Range) ,  
consuming\_flow\_plans (Date\_Range, Logical) -- a List[Flow\_Plan] field of  
model Buffer\_Plan**  
The Flows planned to consume from this Buffer.  
Properties: Export-Only Field

**producing\_flow (Date\_Range) -- a Quantity field of model Buffer\_Plan**  
The total Quantity of Flow\_Plans producing into this Buffer during the given  
Date\_Range.  
Properties: Export-Only Field

**consuming\_flow (Date\_Range) -- a Quantity field of model Buffer\_Plan**  
The total Quantity of Flow\_Plans consuming from this Buffer during the given  
Date\_Range. Note that this will be a negative number.  
Properties: Export-Only Field

**on\_hand\_profile, on\_hand\_profile (Date\_Range) -- a List[Profile\_Quantity]  
field of model Buffer\_Plan**  
A List of the changes to the on\_hand Quantity profile during the specified finite  
Date\_Range.  
Properties: command=True Export-Only Field

**on\_hand, on\_hand (Date) , on\_hand (Date\_Range) , on\_hand (Date\_Range,  
Logical) , on\_hand (Flow\_Plan) -- a Quantity field of model Buffer\_Plan**  
If a Date is passed, this returns the Quantity planned to be on-hand as of that Date. If a  
Period is passed, this returns the smallest Quantity planned to be on-hand during that  
Period. If a Period and a Logical are passed, this returns the smallest Quantity planned  
to be on-hand during the Period if the Logical is True. If it is False, this returns the  
largest Quantity planned during the Period. If a Flow\_Plan is passed, this returns the  
Quantity planned to be on-hand at the completion of the Flow\_Plan. If the Date\_Range  
is passed with the logical argument's value equal to TRUE, it will return smallest

Quantity planned to be on-hand during that date range. If the logical argument's value  
is FALSE, it will return the max Quantity planned to be on-hand during that date  
range. If no argument is passed, this returns the Quantity on-hand on 'on\_hand\_date'. It  
returns 'nonexistent' if the argument Date is earlier than 'on\_hand\_date'. The Quantity  
is converted to the 'unit' of this buffer.

Setting the field sets 'on\_hand\_date' to the argument Date, or to 'now' if no argument,  
and sets this field to report the specified Quantity as on\_hand.

Be careful when setting this to make sure that it is set consistent with the state infor-  
mation of the Operation\_Plans. For example, suppose an Operation\_Plan plans to  
remove 500 units at 11:00, but in reality the units are removed at 10:00. If the drop in  
'on\_hand' is reported (this is set to 500 less), but the units are not reported to have  
'arrived' at the Operation\_Plan, then it will appear that at 11:00 an additional 500 units  
will be removed. Note also that reporting the arrival at the Operation\_Plan will auto-  
matically remove the Quantity from on\_hand.  
Default: 0

Properties: Export-Only Field  
**on\_hand\_date -- a Date field of model Buffer\_Plan**  
The Date at which 'on\_hand' was reported to be accurate.  
Default: infinite\_past

**on\_hand\_lots (Date) -- a List[Lot\_Flow] field of model Buffer\_Plan**  
A list of all Lot\_Flows that make up the on hand amount on the date given.  
Properties: Export-Only Field

**lots -- a list of Lot submodels of model Buffer\_Plan**  
The specific lots planned to flow through this in this Buffer. Note that lots are not  
tracked for STANDARD items.

**flow\_policy -- an extension selector of model Buffer\_Plan**  
This has the same value as 'buffer\_flow\_policy'. This extension defines how the Opera-  
tion Flows placed on this Buffer are planned. The associated fields define the Problems  
that can be caused by the flow of items through this Buffer, and how those Problems  
are resolved. In particular, it defines the relationship between the flows consuming  
from the Buffer and those producing into it. For example, Lot-For-Lot, Fixed, EOQ,  
and POQ inventory policies would be implemented as different flow\_policy's.  
Properties: extension\_same\_as\_model=Buffer Export-Only Field  
Extensions:

HUCKETED\_NESTED\_SORT, PRODUCING\_FLOW\_CALENDAR,  
PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK,  
ON\_HAND\_CALENDAR, ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK,  
FLOW\_LIMIT\_CALENDAR,  
FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK, BASIC,  
BASIC\_FILTER\_AND\_RANK, FIXED\_QUANTITY, MULTIPLE,  
MULTIPLE\_FILTER\_AND\_RANK, FIXED\_QUANTITY\_FENCED,  
FIXED\_TIME.

create\_producing\_operation\_plan (Measure\_Unit, Restriction) - - a  
Operation\_Plan *field of model Buffer\_Plan*

If this Buffer has a 'producing\_operation' designated, then

create\_producing\_operation\_plan will create a new Operation\_Plan of the designated  
Quantity and Restriction, motivated to PRODUCE into this Buffer\_Plan.

Properties: command=True Export-Only Field

create\_consuming\_operation\_plan (Measure\_Unit, Restriction) - - a  
Operation\_Plan *field of model Buffer\_Plan*

If this Buffer has a 'consuming\_operation' designated, then

create\_consuming\_operation\_plan will create a new Operation\_Plan of the designated  
Quantity and Restriction, motivated to CONSUME from this Buffer\_Plan.

Properties: command=True Export-Only Field

problems, problems (Date\_Range) , problems (Problem\_Category) , problems  
(Date\_Range, Problem\_Category) - - a List(Problem) *field of model Buffer\_Plan*  
The Problems detected with this Buffer\_Plan. If passed a Date\_Range, only the Prob-  
lems whose 'dates' overlap are returned. If passed a Problem\_Category, only the Prob-  
lems with that 'category' are returned.

Properties: Export-Only Field

problem\_categories - - a List(Problem\_Category) *field of model Buffer\_Plan*

For each different 'category' of Problem in 'problems', a Problem\_Category is added to  
this list. It gives you the name of the 'category' (in various forms) plus a list of just the  
Problems of that 'category'. These sublists are often much easier to deal with than the  
full 'problems' list.

Properties: Export-Only Field

in\_flow\_plans, in\_flow\_plans (Date\_Range) , in\_flow\_plans (Date\_Range, Logi-  
cal) - - a List(Flow\_Plan) *field of model Buffer\_Plan*  
Obsolete! This field has been renamed 'producing\_flow\_plans' in an effort to have  
more consistent and less ambiguous naming. This field will be eliminated in a future  
release.

Properties: obsolete=True Export-Only Field

out\_flow\_plans, out\_flow\_plans (Date\_Range) , out\_flow\_plans (Date\_Range,  
Logical) - - a List(Flow\_Plan) *field of model Buffer\_Plan*

Obsolete! This field has been renamed 'consuming\_flow\_plans' in an effort to have  
more consistent and less ambiguous naming. This field will be eliminated in a future  
release.

Properties: obsolete=True Export-Only Field

in\_flow (Date\_Range) - - a Quantity *field of model Buffer\_Plan*

Obsolete! This field has been renamed 'producing\_flow' in an effort to have more con-  
sistent and less ambiguous naming. This field will be eliminated in a future release.

Properties: obsolete=True Export-Only Field

out\_flow (Date\_Range) - - a Quantity *field of model Buffer\_Plan*

Obsolete! This field has been renamed 'consuming\_flow' in an effort to have more con-  
sistent and less ambiguous naming. This field will be eliminated in a future release.

Note that this will be a negative number.

Properties: obsolete=True Export-Only Field

owner - - a Site\_Plan *field of model Buffer\_Plan*

The Site\_Plan to which this Buffer\_Plan belongs.

Properties: Export-Only Field

Models	Lot Model
--------	-----------

### 5.1.3.1.3.1 Lot Model

Lot -- *a submodel of model Buffer\_Plan*

A Lot models a quantity of an item that all has common characteristics. In some industries these may be called "batches", "loads", "rolls", "coils", "ingots", "melts", "dye lots", etc. Note that lots are not tracked for STANDARD items.

The Lot model has fields that references these models :  
Configuration, Lot\_Flow, Buffer\_Plan.

These models have a field that is a Lot model :  
Buffer\_Plan, Lot\_Flow.

The key field for this model is name  
This model may be extended with user-defined fields.

name -- *a Symbol field of model Lot*  
The name of this Lot. If not specified, then it is derived from the  
producing\_flow.flow\_plan. The operation\_plan.release\_name is concatenated a  
hyphen and the flow.name. Note that most Lots will not be given unique names until  
they come into existence on the floor. Sometimes, not until they have reached a  
Buffer.  
Default: None -- this is a key field

remark -- *a String field of model Lot*  
Remarks about this particular Lot.  
Default: none

quantity (Date) -- *a Quantity field of model Lot*  
The Quantity of 'configuration' that is in this Lot. This Quantity is converted to the  
unit of the 'owner.buffer'.  
Default: 0

configuration -- *a Configuration field of model Lot*  
The lot-specific information recorded for the items of this Lot. What information is  
provided is specified by the 'spec' extension of the 'item', where the 'item' will be the  
same as 'owner.buffer.item'.  
Default: [unspecified]

Models	Lot Model
--------	-----------

formed -- *a Logical field of model Lot*  
If "true", then this Lot has been formed -- there are 'quantity' of 'configuration' in exist-  
ence. If "false", then this Lot is not yet formed -- it may be planned to exist, or other-  
wise just under consideration.  
Default: false

producing\_flow -- *a Lot\_Flow field of model Lot*  
The Lot\_Flow (and Flow\_Plan) that produces this Lot into this Buffer.  
Properties: Export-Only Field

consuming\_flows -- *a List[Lot\_Flow] field of model Lot*  
The Lot\_Flows (and Flow\_Plans) that consume this Lot from this Buffer.  
Properties: Export-Only Field

supplying\_flow -- *a Lot\_Flow field of model Lot*  
Obsolete! This field has been renamed 'producing\_flow' in an effort to have more con-  
sistent and less ambiguous naming. This field will be eliminated in a future release.  
Properties: obsolete=True Export-Only Field

owner -- *a Buffer\_Plan field of model Lot*  
The Buffer\_Plan through which this Lot flows.  
Properties: Export-Only Field

5.1.3.1.3.2 flow\_policy submodels of model Buffer\_Plan  
5.1.3.1.3.3 Sorted\_Bucket Model

Sorted\_Bucket -- a submodel of model Buffer\_Plan

The buckets that are created by the Buffer's 'horizon' for this Buffer\_Plan.

The Sorted\_Bucket model has fields that references these models :  
Buffer\_Plan.

These models have a field that is a Sorted\_Bucket model :  
BUCKETED\_NESTED\_SORT, BUCKETED\_COMBINED\_SORT.

The key field for this model is dates  
This model may be extended with user-defined fields.

dates - - a Date\_Range field of model Sorted\_Bucket  
The Date\_Range covered by this bucket, as defined by the Buffer's 'horizon':  
Default: None -- this is a key field

ending\_on\_hand - - a Quantity field of model Sorted\_Bucket  
The on\_hand that should be present at the end of this bucket ('dates.end') as computed from the sum of the Buffer 'fixed\_ending\_on\_hand' and the 'per\_next\_ending\_on\_hand' multiplied by the consuming\_flow in the next bucket. A Problem (EXCESS\_ON\_HAND, LOW\_ON\_HAND, or NEGATIVE\_ON\_HAND) will be created if the 'on\_hand' at the end of the bucket is not equal to this 'ending\_on\_hand' Quantity.

This field, though typically computed, is settable. Setting 'ending\_on\_hand' will also set 'lock\_ending\_on\_hand' to 'true' -- which will prevent this field from being recomputed from the Buffer fields -- it will remain the value set into it.

This allows the user to easily manipulate the target ending\_on\_hand value for a particular month, without needing to modify the Buffer parameters. This is often used to "release" some of the planned inventory in the near-term if supply is not sufficient to cover demand.

Default: computed from the Buffer fields

lock\_ending\_on\_hand - - a Logical field of model Sorted\_Bucket  
If "false", then the 'ending\_on\_hand' is computed from the sum of the Buffer 'fixed\_ending\_on\_hand' and the 'per\_next\_ending\_on\_hand' multiplied by the consuming\_flow in the next bucket. If "true", then the current 'ending\_on\_hand' setting will be used, overriding the setting that would normally be computed based upon the fields in the Buffer.

This allows the user to easily manipulate the target ending\_on\_hand value for a particular month, without needing to modify the Buffer parameters. This is often used to "release" some of the planned inventory in the near-term if supply is not sufficient to cover demand.  
Default: false

producing\_flow\_plans - - a List(Flow\_Plan) field of model Sorted\_Bucket  
The Flow\_Plans that produce into this Buffer\_Plan during this Sort\_Bucket.

If these Flow\_Plans were motivated by this Buffer\_Plan, then they will correspond to the specified 'criteria' that are marked 'produce\_separately'.  
Properties: Export-Only Field

consuming\_flow\_plans - - a List(Flow\_Plan) field of model Sorted\_Bucket  
The Flow\_Plans that consume from this Buffer\_Plan during this Sort\_Bucket, sorted according to the Buffer 'flow\_policy' and the specified sort 'criteria'.  
Properties: Export-Only Field

owner - - a Buffer\_Plan field of model Sorted\_Bucket

Properties: Export-Only Field  
5.1.3.1.4 Resource\_Plan Model

Resource\_Plan -- a submodel of model Site\_Plan

A Resource models the machines, tools, fixtures, labor, and other things that are used by Operations in transforming items. The capacity of a Plan to perform Operations is modeled by the Resources.

Resource\_State models the "current" state of a Resource: its current setup, its current location, its current operating efficiency (0% means its down), and its current activity.

The model has selectors:  
efficiency, load\_policy, size, maintenance.

The Resource\_Plan model has fields that references these models :  
Resource, Location, Skill, Site\_Plan.

These models have a field that is a Resource\_Plan model :  
Plan, Resource, LINK, Load\_Plan, TRANSIT, SETUP, SKILL, MAINTENANCE, Consolidation, UNCONSOLIDATED, UNCOORDINATED, CONSOLIDATION\_OVERSIZE, OVERLOAD, OVERTIME, OVERSIZE, BUCKET\_OVERSIZE, UNDERLOAD, CUMULATIVE\_OVERLOAD.

The key field for this model is resource  
This model may be extended with user-defined fields.

resource - - a Resource field of model Resource\_Plan  
The Resource for which this Resource\_Plan is planning.  
Default: None -- this is a key field

remark - - a String field of model Resource\_Plan  
Comments about this specific plan.  
Default: none

efficiency\_profile (Date\_Range) - - a List(Profile\_Percentage) field of model Resource\_Plan

A List of all changes in the efficiency profile during the specified finite Date\_Range. This is the resultant efficiency profile generated by combining efficiency specified by 'resource' efficiency extension and all the efficiency changes made during planning by incurring extra cost. Note, that this is just the resource efficiency ignoring the efficiency at particular skill.

Properties: Export-Only Field

efficiency\_average (Date\_Range) - - a Percentage field of model Resource\_Plan  
The average efficiency during a particular Date\_Range (or at a particular Date).

Properties: Export-Only Field

efficiency\_profile\_at\_skill (Skill, Date\_Range) - - a List(Profile\_Percentage) field of model Resource\_Plan

A List of all changes in the efficiency profile during the specified finite Date\_Range. Values of this profile combines efficiencies specified by 'resource' efficiency extension, plus the the efficiency changes made during planning, by incurring extra cost and efficiency of 'resource' at performing specified 'skill'. If the skill passed in as an argument is unspecified, then skill efficiency will be ignored. If the 'resource' does not belong to the skill group specified by the 'skill', it will return 0 efficiency.

Properties: Export-Only Field

efficiency\_average\_at\_skill (Skill, Date\_Range) - - a Percentage field of model Resource\_Plan

The average efficiency during a particular Date\_Range (or at a particular Date). Average efficiency value of this profile combines efficiencies specified by 'resource' efficiency extension, plus the the efficiency changes made during planning, by incurring extra cost and efficiency of 'resource' at performing specified 'skill'.

Properties: Export-Only Field

available\_time (Date\_Range) - - a Time field of model Resource\_Plan

The total actual hours that this Resource is planned to be available (efficiency > 0%) during a particular Date\_Range. These are actual hours, not efficiency-adjusted "standard hours". See the 'capacity' efficiency-adjusted standard hours.

Properties: Export-Only Field

capacity (Date\_Range) - - a Time field of model Resource\_Plan

The total efficiency-adjusted "standard hours" of capacity available from this Resource during a particular Date\_Range. This is the "standard hours" of load that can be handled by this Resource during the given dates (see 'load\_std\_time').

Mathematically, 'capacity(pd)' is the same as '(pd.end - pd.start) \* efficiency\_average(pd)', which is equivalent to the traditional definition of capacity, the available\_time \* the efficiency during that time. For actual hours of availability, see 'available\_time'.

Properties: Export-Only Field

efficiency - - an extension selector of model Resource\_Plan

This has the same value as 'resource.efficiency'. This extension defines how the availability, capacity, and efficiency of this Resource is modeled. The associated fields describe the efficiency of this Resource over time.

Properties: extension\_same\_as\_model=Resource Export-Only Field

**load\_plans, load\_plans (Date\_Range) - - a List[Load\_Plan] field of model Resource\_Plan**

The Loads that are planned on this Resource, excluding self-imposed setup and maintenance. If passed a Date\_Range, only the Load\_Plans that overlap that Date\_Range are returned.

Properties: Export-Only Field

**load\_time (Date\_Range) - - a Time field of model Resource\_Plan**

The total actual hours of load that is planned on this Resource during a particular Date\_Range. This excludes self-imposed setup\_change and maintenance time. For comparison, load\_time(pd) is comparable to available\_time(pd). See load\_std\_time for total "standard hours" of load, which is comparable to capacity.

Properties: Export-Only Field

**load\_std\_time (Date\_Range) - - a Time field of model Resource\_Plan**

The total "standard hours" of load that is planned on this Resource during a particular Date\_Range. This excludes self-imposed setup\_change and maintenance time. For comparison, load\_std\_time(pd) is comparable to capacity(pd). See load\_time for total actual hours of load, which is comparable to available\_time.

Properties: Export-Only Field

**load\_policy - - an extension selector of model Resource\_Plan**

This has the same value as resource.load\_policy. This extension defines how the Operation Loads placed on this Resource are planned. The associated fields describe the rules and restrictions on the Loads and the Problems that could occur due to usage of this Resource's capacity.

Properties: extension\_same\_as\_model=Resource Export-Only Field

Extensions:

**SIMPLE\_CONSOLIDATION, INFINITE\_USE, EXCLUSIVE\_USE, SHARED\_USE.**

**balance\_time (Date\_Range, Logical, Logical, Logical) - - a Void field of model Resource\_Plan**

Apply the intelligence of the load\_policy to move the load\_plans so as to balance the load\_time with the available\_time in the specified Date\_Range. The three Logical arguments specify whether or not to try moving earlier, moving later, and/or moving off to an alternate, respectively.

Properties: command=True Export-Only Field

**balance\_std\_time (Date\_Range, Logical, Logical, Logical) - - a Void field of model Resource\_Plan**

Apply the intelligence of the load\_policy to move the load\_plans so as to balance the load\_std\_time with the capacity in the specified Date\_Range. The three Logical arguments specify whether or not to try moving earlier, moving later, and/or moving off to an alternate, respectively.

Properties: command=True Export-Only Field

**move (Load\_Plan, Logical, Logical, Logical) - - a Void field of model Resource\_Plan**

Apply the intelligence of the load\_policy to move the load\_plans so as to balance the load\_std\_time with the capacity in the specified Date\_Range. The three Logical arguments specify whether or not to try moving earlier, moving later, and/or moving off to an alternate, respectively.

Properties: command=True Export-Only Field

**previous\_gap (Load\_Plan) - - a Date field of model Resource\_Plan**

The Date of the latest gap in load before the argument load\_plan's end Date, excluding the load due to the argument load\_plan itself. There may not be sufficient gap to perform the entire argument load\_plan without Problem, but at least a portion at the end of the load\_plan can be performed without Problem.

Properties: Export-Only Field

**next\_gap (Load\_Plan) - - a Date field of model Resource\_Plan**

The Date of the earliest gap in load after the argument load\_plan's start Date, excluding the load due to the argument load\_plan itself. There may not be sufficient gap to perform the entire argument load\_plan without Problem, but at least a portion at the start of the load\_plan can be performed without Problem.

Properties: Export-Only Field

**size - - an extension selector of model Resource\_Plan**

This has the same value as resource.size. It defines the size limits on the loads that can be placed on this Resource. The associated fields describe the size (volume, weight) limits of this Resource over time. The size fields indicate how much Load can be handled at once. For example, if the size limit is "2 tons", then up to 2 tons of Loads may be simultaneously processed.

Properties: extension\_same\_as\_model=Resource Export-Only Field

**size\_profile (Date\_Range)** - - *a List(Profile\_Quantity) field of model Resource\_Plan*

A List of all changes in the available size profile during the specified finite

Date\_Range.

Properties: Export-Only Field

**size\_usage\_profile (Date\_Range)** - - *a List(Profile\_Quantity) field of model Resource\_Plan*

A List of all changes in the profile of usage of size (the size of Loads) during the specified finite Date\_Range.

Properties: Export-Only Field

**available\_sized\_time (Date\_Range)** - - *a Quantity field of model Resource\_Plan*

The total actual hours that this Resource is planned to be available (efficiency > 0%) during a particular Date\_Range, multiplied by the 'size' that is available during that time.

This is based on actual hours, not efficiency-adjusted "standard hours". See the 'sized\_capacity' for efficiency-adjusted standard hours.

Properties: Export-Only Field

**sized\_capacity (Date\_Range)** - - *a Quantity field of model Resource\_Plan*

The total efficiency-adjusted "standard hours" of capacity available from this Resource during a particular Date\_Range, multiplied by the 'size' that is available during that time.

For actual hours of availability, see 'available\_time'; for standard hours, see 'capacity'; and for sized hours, see 'available\_sized\_time'.

Properties: Export-Only Field

**load\_sized\_time (Date\_Range)** - - *a Quantity field of model Resource\_Plan*

The total actual hours of load that is planned on this Resource during a particular Date\_Range, multiplied by the 'size' of that load over that time. This excludes self-imposed setup\_change and maintenance time. For comparison, 'load\_sized\_time(pd)' is comparable to 'available\_sized\_time(pd)'. See 'load\_sized\_std\_time' for a similar sized measure of load based on "standard hours", which is comparable to 'sized\_capacity'.

Properties: Export-Only Field

**load\_sized\_std\_time (Date\_Range)** - - *a Quantity field of model Resource\_Plan*

The total "standard hours" of load that is planned on this Resource during a particular Date\_Range, multiplied by the 'size' of that load over that time. This excludes self-imposed setup\_change and maintenance time. For comparison,

'load\_sized\_std\_time(pd)' is comparable to 'sized\_capacity(pd)'. See 'load\_sized\_time' for a similar sized measure of load based on total actual hours of load, which is comparable to 'available\_sized\_time'.

Properties: Export-Only Field

**balance\_sized\_time (Date\_Range, Logical, Logical, Logical)** - - *a Void field of model Resource\_Plan*

Apply the intelligence of the 'load\_policy' to move the 'load\_plans' so as to balance the 'load\_sized\_time' with the 'available\_sized\_time' in the specified Date\_Range. The three Logical arguments specify whether or not to try moving earlier, moving later, and/or moving off to an alternate, respectively.

Properties: command=True Export-Only Field

**balance\_sized\_std\_time (Date\_Range, Logical, Logical, Logical)** - - *a Void field of model Resource\_Plan*

Apply the intelligence of the 'load\_policy' to move the 'load\_plans' so as to balance the 'load\_sized\_std\_time' with the 'sized\_capacity' in the specified Date\_Range. The three Logical arguments specify whether or not to try moving earlier, moving later, and/or moving off to an alternate, respectively.

Properties: command=True Export-Only Field

**maintenance** - - *an extension selector of model Resource\_Plan*

Defines how maintenance is specified for this Resource. The associated fields describe when maintenance should be performed and what Operation is used to perform it. It may describe both major and minor maintenance Operations. It may base timing either total time, loaded time, setup time, regular calendar intervals, or many other criteria.

Properties: extension\_same\_as\_model=Resource Export-Only Field

**problems, problems (Date\_Range)** , **problems (Problem\_Category)** , **problems (Date\_Range, Problem\_Category)** - - *a List(Problem) field of model Resource\_Plan*

The Problems detected with this Resource\_Plan. If passed a Date\_Range, only the Problems whose 'dates' overlap are returned. If passed a Problem\_Category, only the Problems with that 'category' are returned.

Properties: Export-Only Field



Models	Resource_Plan Model
--------	---------------------

**problem\_categories** - - *a List{Problem\_Category} field of model Resource\_Plan*  
 For each different category of Problem in problems, a Problem\_Category is added to this list. It gives you the name of the category (in various forms) plus a list of just the Problems of that category. These sublists are often much easier to deal with than the full problems list.  
 Properties: Export-Only Field

**problem\_time (Date\_Range)** - - *a Time field of model Resource\_Plan*  
 The total actual hours of Problems with the load that is planned on this Resource during a particular Date\_Range. For comparison, problem\_time(pd) is comparable to available\_time(pd). See problem\_std\_time for total "standard hours" of Problems, which is comparable to capacity.  
 Properties: Export-Only Field

**problem\_std\_time (Date\_Range)** - - *a Time field of model Resource\_Plan*  
 The total "standard hours" of Problems with the load that is planned on this Resource during a particular Date\_Range. For comparison, load\_std\_time(pd) is comparable to capacity(pd). See load\_time for total actual hours of load, which is comparable to available\_time.  
 Properties: Export-Only Field

**problem\_sized\_time (Date\_Range)** - - *a Quantity field of model Resource\_Plan*  
 The total actual hours of Problems with the load that is planned on this Resource during a particular Date\_Range, multiplied by the size of those Problems over that time. For comparison, problem\_sized\_time(pd) is comparable to available\_sized\_time(pd). See problem\_sized\_std\_time for a similar sized measure of load based on "standard hours", which is comparable to sized\_capacity.  
 Properties: Export-Only Field

**problem\_sized\_std\_time (Date\_Range)** - - *a Quantity field of model Resource\_Plan*  
 The total "standard hours" of Problems with the load that is planned on this Resource during a particular Date\_Range, multiplied by the size of those Problems over that time. For comparison, problem\_sized\_std\_time(pd) is comparable to sized\_capacity(pd). See problem\_sized\_time for a similar sized measure of load based on total actual hours of load, which is comparable to available\_sized\_time.  
 Properties: Export-Only Field

Models	Resource_Plan Model
--------	---------------------

**resolve (Problem, Logical, Logical, Logical)** - - *a Void field of model Resource\_Plan*  
 Resolve the Problems in the argument List, or in the specified Date\_Range. The three Logical arguments specify whether or not to try moving earlier, moving later, and/or moving off to an alternate, respectively.  
 Properties: command=True Export-Only Field

**owner** - - *a Site\_Plan field of model Resource\_Plan*  
 The Site\_Plan for which this Resource\_Plan is a component.  
 Properties: Export-Only Field

Model	load_policy submodels of model Resource_Plan
-------	--

### 5.1.3.1.4.1 load\_policy submodels of model Resource\_Plan

#### 5.1.3.1.4.2 Consolidation Model

Consolidation -- a submodel of model Resource\_Plan

Consolidation is sub-model of resource\_plan created to group load\_plans together. As of today, they are supported for SIMPLE\_CONSOLIDATION load\_policy only.

The Consolidation model has fields that references these models :  
Resource\_Plan.

These models have a field that is a Consolidation model :  
SIMPLE\_CONSOLIDATION, UNCOORDINATED,  
CONSOLIDATION\_OVERSIZE, CONSOLIDATION\_UNDERSIZE.

The key field for this model is name  
This model may be extended with user-defined fields.

name -- a Symbol field of model Consolidation  
The name of this Consolidation.  
Default: None -- this is a key field

operation\_plans -- a List(Operation\_Plan) field of model Consolidation  
The list of Operation\_Plans in this Consolidation.  
Properties: Export-Only Field

inc (Operation\_Plan) -- a Void field of model Consolidation  
Add an opplan to this consolidation Adding an opplan can create UNCOORDINATED or  
CONSOLIDATION\_OVERSIZE problem.  
Properties: command=True Export-Only Field

dec (Operation\_Plan) -- a Void field of model Consolidation  
remove an opplan from the consolidation Removing an opplan can cause UNCONSOLI-  
DATED and/or CONSOLIDATION\_UNDERSIZE problem.  
Properties: command=True Export-Only Field

owner -- a Resource\_Plan field of model Consolidation  
The owner of this Consolidation  
Properties: Export-Only Field

#### 5.1.3.1.5 Operation\_Plan Model

Model	Operation_Plan Model
-------	----------------------

Operation\_Plan -- a submodel of model Site\_Plan

The plan for performing an Operation. It includes start and end Dates for the activi-  
ties, the Resources that will be loaded and when, the Buffers that will be consumed  
from or produced into and when, and all other information specific to a particular  
Operation.

If the Operation\_Plan has been released to be performed, then it will be given a  
'release\_name'. At that point, changes to the plan are only made due to feasibility  
Problems related to state information. State information is recorded in the  
Operation\_Plan, but may come indirectly from Operation\_State models.

The state fields are designed to support the "minimal transaction" philosophies, such  
as Just-In-Time (JIT). As time marches forward, we simply assume everything is  
going according to the Plan, unless told otherwise in the form of new state settings.

For similar reasons, plus the fact that state information often comes from other tools,  
we do not demand the user specify the Operation\_Plan or even a 'release\_name' in  
order to report state information. See the Operation\_State model for that functionality.

The model has selectors:  
motive, process.

The Operation\_Plan model has these submodels :  
Load\_Plan, Flow\_Plan.

The Operation\_Plan model has fields that references these models :  
Operation, Load\_Plan, Flow\_Plan, Operation\_Plan, Site\_Plan.

These models have a field that is a Operation\_Plan model :  
(Operation\_Plan, Buffer\_Plan, LINK, Operation\_State, Item\_Request,  
Load\_Plan, Flow\_Plan, Item\_Acceptance, Item\_Promise,  
REQUEST\_NOT\_PLANNED, REQUEST\_PLANNED\_LATE,  
REQUEST\_PLANNED\_EARLY, REQUEST\_PLANNED\_SHORT,  
REQUEST\_PLANNED\_EXCESS, PROMISE\_NOT\_PLANNED,  
PROMISE\_PLANNED\_LATE, PROMISE\_PLANNED\_EARLY,  
PROMISE\_PLANNED\_SHORT, PROMISE\_PLANNED\_EXCESS,  
ACCEPTANCE\_NOT\_PLANNED, ACCEPTANCE\_PLANNED\_LATE,  
ACCEPTANCE\_PLANNED\_EARLY, ACCEPTANCE\_PLANNED\_SHORT,  
ACCEPTANCE\_PLANNED\_EXCESS, PRECEDENCE,  
OVER\_RESTRICTION, EXPEDITED,  
PLANNED\_BEFORE\_CURRENT, UNRELEASED, NEEDS\_RELEASE,

INCONSISTENT\_OPPLAN, SUPPLY\_PLANNED\_LATE, SUPPLY\_PLANNED\_EARLY, SUPPLY\_PLANNED\_SHORT, SUPPLY\_PLANNED\_EXCESS, SUPPLY\_PROMISED\_LATE, SUPPLY\_PROMISED\_EARLY, SUPPLY\_PROMISED\_SHORT, SUPPLY\_PROMISED\_EXCESS.

The key field for this model is operation  
This model may be extended with user-defined fields.

operation - - a *Operation field of model Operation\_Plan*  
The Operation being planned by this Operation\_Plan. The Operation defines what is to be done by this Operation\_Plan.

Default: None -- this is a key field

remark - - a *String field of model Operation\_Plan*  
Comments about this specific plan to perform this 'operation'.  
Default: none

rank - - a *Number field of model Operation\_Plan*  
The 'rank' of this Operation\_Plan, which is a relative measure of importance, priority, or significance. Its value can be dynamic (changed by the planning algorithms); its effects can be dynamic (different algorithms and different Strategies may treat ranks differently); and overall its semantic is user-controllable. This is used to lock Operation\_Plans during strategy runs when used in conjunction with Strategy\_Lock model.

Rank can be set on a top Operation\_Plan at the time of its creation, using the Plan's set\_default\_operation\_plan\_rank expression.  
Default: 0

released - - a *Logical field of model Operation\_Plan*  
If "true", then this Operation\_Plan has been released to be performed. Once released, it is subject to greater constraints. Its motive and quantity will not be changed, and it will be executed as soon as feasible. The material assigned to it is considered "allocated" and will not be "re-allocated". Of course, users may change anything about a released Operation\_Plan they desire. When the Operation\_Plan is for an operation which is part of an operation hierarchy, (it has sub-operations, or it is itself a sub-operation, or both), the entire hierarchy is released together.  
Default: false

release\_name - - a *Symbol field of model Operation\_Plan*  
The 'release\_name' used to identify an Operation\_Plan when it is being processed. This is often called an "order id". This need not be used -- for example, repetitive manufacturers often do not need to bother with a 'release\_name'. But even in that case, a 'release\_name' will be generated when 'released' is set "true" (see the description of the release\_name\_expression field). If the Operation\_Plan is was not released when the release\_name is set, the Operation\_Plan will be released. As with released Operation\_Plans, if the Operation\_Plan is part of a hierarchy, the release\_name refers to the entire hierarchy.  
Default: none

release\_fence\_date - - a *Date field of model Operation\_Plan*  
Returns the date this op\_plans release\_fence becomes effective.

Default: 0

Properties: Export-Only Field

release\_soon\_fence\_date - - a *Date field of model Operation\_Plan*  
Returns the date this op\_plans release\_soon\_fence becomes effective.

Default: 0

Properties: Export-Only Field

use\_alternate (Operation\_Plan), use\_alternate (Operation\_Plan, Operation) - - a *Void field of model Operation\_Plan*  
Replan the Operation\_Plan to try to use a different sub\_operation\_plan.  
Properties: command=True Export-Only Field

motive - - an *extension selector of model Operation\_Plan*  
The purpose this Operation\_Plan was created. It is one of: PRODUCE (produce 'quantity' units into a Buffer), CONSUME (consume 'quantity' units from a Buffer), DELIVER (deliver 'quantity' units to an address), RECEIVE (receive 'quantity' units from an address).

This field is read-only.

Properties: Export-Only Field

Extensions:

PRODUCE, CONSUME, DELIVER, RECEIVE,

units - - a *Number field of model Operation\_Plan*  
The number of units of this Operation planned. Note that, due to yields, this Quantity may vary as the Operation\_Plan moves, even when the 'motive' remains the same.

This is always unitless, of course -- the Number of units of this Operation. See 'quantity' for the same value returned in the Operation's 'preferred\_measure'.  
Default: computed from the Operation

**quantity** - - *a Quantity field of model Operation\_Plan*  
The units of this Operation planned. Note that, due to yields, this Quantity may vary as the Operation\_Plan moves, even when the 'motive' remains the same.

This Quantity is converted to the unit of this Operation.  
Default: computed from the Operation

**std\_time** - - *a Time field of model Operation\_Plan*  
The standard Time required by this Operation\_Plan to fulfill the 'motive'. This is computed by multiplying the standard Time computed by the 'operation' with the 'expedite' Percentage.

Setting this field actually sets the 'expedite' field appropriately to give the specified result. An EXPEDITED Problem will be raised if 'expedite' is set to other than 100%.  
Default: computed from the Operation

**expedite** - - *a Percentage field of model Operation\_Plan*  
Expedite values less than 100% specify a reduction in time for a particular operation plan with respect to the standard time specified by the operation. When the expedite value is greater than 100% then the operation plan takes longer. To define an operation that takes only half as long as the standard, set expedite to be 50%. Stated another way: the expedite number is 100% divided by the speed increase you desire for this particular operation plan. For 3 times faster than normal, set expedite to 33%.

The 'std\_time' member will reflect the operation's standard Time multiplied by the expedite Percentage. If the 'operation' specifies the standard Time to be 8 hours and the 'expedite' Percentage is set to "25%", then the 'std\_time' will be 2 hours.

The actual time for the operation will reflect the resource loading efficiency (assuming that the process actually loads a resource -- some don't).  
Default: 100%

**dates** - - *a Date\_Range field of model Operation\_Plan*  
The Date\_Range during which this Operation\_Plan is planned to be performed. So, 'dates.start' is the start Date; 'dates.end' is the end Date. And 'dates.time' is the actual duration of this Operation\_Plan. It is primarily affected by the 'efficiency' of each of the Resources this Operation\_Plan is loading. The minimum efficiency of the Resources at any given Date is the efficiency of this Operation\_Plan at that Date.

Properties: Export-Only Field

**hint** - - *a Restriction field of model Operation\_Plan*  
A temporary Restriction on this Operation\_Plan. Setting this causes the Operation\_Plan to be moved accordingly (if possible). For example, setting it to "start after 97-05-01 12:00" will move the planned 'dates' to "97-05-01 12:00". If it is already after "97-05-01 12:00", then it will not move. Note that the engine will accept second granularity, but requires at least minute granularity.

The default value is "[unspecified]" (however, the engine will accept [] and change it to [unspecified]).

The hint can have one of the following forms: "start before <date>", "start after <date>", "end before <date>", "end after <date>", "first in <period>", "last in <period>", or "[unspecified]".

The "start before <date>" and "start after <date>" actually executes as "start at" and "end before". The "end before" and "end after" both execute as "end at".

The "first in <period>" option is actually a choice between "end first in <date\_range>" and "start first in <date\_range>". If the user types "first in <date\_range>", it automatically changes to "start first in <date\_range>". The "start first in <date\_range>" sets the start of the operation plan as early in the date\_range as possible. The "end first in <date\_range>" sets the end of the operation plan as early in the date\_range as possible.

The "last in <period>" option is actually a choice between "end last in <date\_range>" and "start last in <date\_range>". If the user types "last in <date\_range>", it automatically changes to "end last in <date\_range>". The "start last in <date\_range>" sets the start of the operation plan as late in the date\_range as possible. The "end last in <date\_range>" sets the end of the operation plan as late in the date\_range as possible.

Note that this setting is temporary, but once the engine attempts to follow the hint, the engine never again attempts to follow the hint until the user sets (types) it again.  
Default: empty (within the infinite Date\_Range)

**locked\_as\_planned** - - *a Logical field of model Operation\_Plan*  
A flag which indicates if this Operation\_Plan cannot be edited (moved, resized, etc.). Attempts to replan a locked\_as\_planned operation plan will fail. If this results in a situation where a locked\_as\_planned operation plan causes the plan to be inconsistent due to an edit of a model, an INCONSISTENT\_OPPLAN problem is raised. See the description of the INCONSISTENT\_OPPLAN problem for more detailed information.

Models	Operation_Plan Model
--------	----------------------

Default: FALSE

load\_plans - - *a list of Load\_Plan submodels of model Operation\_Plan*

These load\_plans specify the Resources to be loaded for the duration of this Operation.

flow\_plans - - *a list of Flow\_Plan submodels of model Operation\_Plan*  
The Item Flow\_Plan generated for the duration of this Operation.

all\_consuming\_flow\_plans - - *a List(Flow\_Plan) field of model Operation\_Plan*

Return all consuming Flow\_Plan in this Operation\_Plan and its sub\_operation\_plans.

Properties: Export-Only Field

all\_producing\_flow\_plans - - *a List(Flow\_Plan) field of model Operation\_Plan*

Return all producing Flow\_Plan in this Operation\_Plan and its sub\_operation\_plans.

Properties: Export-Only Field

sub\_operation\_plans - - *a List(Operation\_Plan) field of model Operation\_Plan*

The Operation\_Plan that model smaller portions/phases of this Operation\_Plan.

Properties: Export-Only Field

super\_operation\_plan - - *a Operation\_Plan field of model Operation\_Plan*

The routing, alternates, or other containing Operation\_Plan of which this

Operation\_Plan is a part. This may be a coordinating Operation\_Plan that does not really correspond to the Operations in the Site.

This may be "nonexistent" if this Operation was directly planned by either 'buffer\_plan' or 'item\_promise'.

Properties: Export-Only Field

top\_operation\_plan - - *a Operation\_Plan field of model Operation\_Plan*

The topmost Operation\_Plan containing this Operation\_Plan. It is equivalent to calling 'super\_operation\_plan' repeatedly until it returns nonexistent.

Properties: Export-Only Field

process - - *an extension selector of model Operation\_Plan*

This is the same as 'operation.process'. It defines the fields that are specific to this Operation's process.

Properties: extension\_same\_as\_model=Operation Export-Only Field

Extensions:

Models	Operation_Plan Model
--------	----------------------

ALTERNATES\_PRIMARY, ALTERNATES\_PROPORTIONAL, EFFECTIVE\_CALENDAR, DELAY\_ONLY\_FIXED, DELAY\_ONLY\_BASIC, BASIC\_CALENDARS, FIXED\_TIME, TIME\_MULTIPLE, BASIC, BASIC\_DELAYED, REQUEST\_FIXED, REQUEST\_FIXED\_WITH\_ANALYSIS, ROUTING,

problems, problems (Date\_Range) , problems (Problem\_Category) , problems (Date\_Range, Problem\_Category) - - *a List(Problem) field of model Operation\_Plan*

The Problems detected with this Operation\_Plan. If passed a Date\_Range, only the

Problems whose 'dates' overlap are returned. If passed a Problem\_Category, only the Problems with that 'category' are returned.

Properties: Export-Only Field

problem\_categories - - *a List(Problem\_Category) field of model Operation\_Plan*

For each different 'category' of Problem in 'problems', a Problem\_Category is added to this list. It gives you the name of the 'category' (in various forms) plus a list of just the Problems of that 'category'. These sublists are often much easier to deal with than the full 'problems' list.

Properties: Export-Only Field

split (Item, Quantity, Logical), split (Quantity) , split (Quantity, Restriction) -

- *a Operation\_Plan field of model Operation\_Plan*

Split will allow users to split this operation plan in to two considering given parameters. It will return the newly created second operation plan as a result of split.

Properties: command=True Export-Only Field

owner - - *a Site\_Plan field of model Operation\_Plan*

Properties: Export-Only Field

### 5.1.3.1.5.1 Load\_Plan Model

#### Load\_Plan -- a submodel of model Operation\_Plan

The Resource\_Plan planned to be loaded by the 'owner' Operation\_Plan, for placement of a particular load on a Resource, typically by an Operation or Buffer. A Load has start and end dates (and thus duration) and a Quantity. It knows the Resource it is loading, as well as the Operation or Buffer placing the load.

The Load\_Plan model has fields that references these models :  
Load, Resource\_Plan, Operation\_Plan.

These models have a field that is a Load\_Plan model :  
Operation\_Plan.

The key field for this model is load

load -- a Load field of model Load\_Plan

The Load of the 'owner' Operation\_Plan for which this is a plan. The Load specifies a Skill which could include many Resources. This Load\_Plan specifies which of those Resources will be loaded.

Default: None -- this is a key field  
Properties: Export-Only Field

resource\_plan -- a Resource\_Plan field of model Load\_Plan

The Resource that is being loaded. When this is set by the user, the hint is set to "ON" to indicate the user is temporarily restricting that choice, as a hint to the automated algorithms. To make this setting stick, set 'lock' to "true".  
Default: [unspecified]

dates -- a Date\_Range field of model Load\_Plan

The dates during which this Load is planned to occur. These 'dates' will be within the 'owner' Operation\_Plan's 'dates'.  
Properties: Export-Only Field

hint -- a Restriction field of model Load\_Plan

A temporary Restriction on this Operation\_Plan. Setting this causes the Operation\_Plan to be moved accordingly (if possible). For example, setting it to "start after 05-01" will move the planned 'dates' such that this Load\_Plan occurs on or after "05-01".

See the hint field in the Operation\_Plan for more information.

Note that setting this hint will generally set the 'owner:hint', though to possibly a somewhat adjusted value designed to reflect the difference between restricting the timing of the Load\_Plan rather than the whole Operation\_Plan.

Note that relaxing the Restriction will not cause the Operation\_Plan to move -- it only moves if it does not already satisfy the Restriction. Thus, setting the hint back to unrestricted will not cause the Operation\_Plan to move back to where it was.  
Default: empty (within the infinite Date\_Range)

hint\_on -- a Logical field of model Load\_Plan

This is a temporary specification that the 'owner' Operation\_Plan should continue to load this Resource\_Plan. If "true", then effectively 'locked' is set temporarily, such that it will not be moved to an alternate. Note, though, unlike 'locked', this value can be ignored and/or reset.  
Default: false

lock\_on -- a Logical field of model Load\_Plan

If "true", then the 'owner' Operation\_Plan is locked onto this 'resource\_plan'. It cannot be moved to an alternate.  
Default: false

use\_alternate, use\_alternate (Resource\_Plan) -- a Void field of model Load\_Plan

Replan the 'owner' Operation\_Plan to use a different Resource\_Plan, if possible. If a Resource\_Plan is specified, then switch to that Resource\_Plan, if possible, and set hint\_on to "true". When given a Resource\_Plan argument, it is the same as setting 'resource\_plan' field.

Properties: command=True Export-Only Field

owner -- a Operation\_Plan field of model Load\_Plan

The Operation\_Plan placing this Load\_Plan on 'resource'.  
Properties: Export-Only Field

### 5.1.3.1.5.2 Flow\_Plan Model

**Flow\_Plan** -- *a submodel of model Operation\_Plan*

The plan for flow of items between Buffers and Operations. A Flow\_Plan has a Quantity, a Buffer, an Operation, and a direction ('produced' or not).

The Flow\_Plan model has these submodels :  
*Lot\_Flow.*

The Flow\_Plan model has fields that references these models :  
*Flow, Buffer\_Plan, Lot\_Flow, Operation\_Plan.*

These models have a field that is a Flow\_Plan model :  
*Operation\_Plan, Lot\_Flow.*

The key field for this model is flow

**flow** -- *a Flow field of model Flow\_Plan*

The Flow of the 'owner' operation for which this is a plan. The Flow specifies how much flows between the Buffer and the Operation. This Flow\_Plan specifies how much will flow for this particular Operation\_Plan.

Default: None -- this is a key field  
 Properties: Export-Only Field

**buffer\_plan** -- *a Buffer\_Plan field of model Flow\_Plan*

The Buffer\_Plan that is being produced into or consumed from by the 'owner' Operation\_Plan.

Properties: Export-Only Field

**produced** -- *a Logical field of model Flow\_Plan*

If "true", then the 'owner' Operation is producing 'quantity' items into the buffer. If "false", then the 'owner' Operation is consuming 'quantity' items from the buffer. This is specified by the flow's usage, policy and related fields.

Properties: Export-Only Field

**dates** -- *a Date\_Range field of model Flow\_Plan*

The dates during which this Flow is planned to occur. These 'dates' will be within the 'owner' Operation\_Plan's dates.  
 Properties: Export-Only Field

quantity, quantity (Date) , quantity (Date\_Range) -- *a Quantity field of model Flow\_Plan*

The Quantity that is being consumed by the 'owner' Operation\_Plan. Note that if the field 'produced' is yes, then this Quantity will be negative -- this field is always the amount consumed from the Buffer.

If no argument is passed, then this returns the quantity consumed during the entire duration of the operation ; If a single Date is passed, then it will give the Quantity consumed by that Date. If a pair of Dates is passed, then it will give the Quantity consumed between those two Dates. (Passing one Date is equivalent to passing the operation's start\_date and that Date; passing nothing is equivalent to passing the operation's start and end dates.)

The Quantity is converted to the 'unit' of the buffer\_plan.buffer.  
 Default: 0

**lots** -- *a list of Lot\_Flow submodels of model Flow\_Plan*  
 The lots being consumed from or produced into a Buffer.

**upstream\_flow\_plans** -- *a List(Flow\_Plan) field of model Flow\_Plan*

A list of all Flow\_Plans that feed this one, either through a Buffer\_Plan or an Operation\_Plan. If this is a 'produced' Flow\_Plan, then this is a list of the Flow\_Plans consumed by the Operation\_Plan that will produce this Flow\_Plan. If this is not a 'produced' Flow\_Plan, then this is a list of the Flow\_Plans that will produce the items that this Flow\_Plan will consume from the Buffer\_Plan.

Properties: command=True Export-Only Field

**downstream\_flow\_plans** -- *a List(Flow\_Plan) field of model Flow\_Plan*

A list of all Flow\_Plans that are fed by this one, either through a Buffer\_Plan or an Operation\_Plan. If this is a 'produced' Flow\_Plan, then this is a list of the Flow\_Plans that will consume the items from the Buffer\_Plan that this Flow\_Plan produces. If this is not a 'produced' Flow\_Plan, then this is a list of the Flow\_Plans that are produced by the Operation\_Plan that consumes this Flow\_Plan.

Properties: command=True Export-Only Field

**owner** -- *a Operation\_Plan field of model Flow\_Plan*

The Operation\_Plan placing this Flow\_Plan on 'buffer'.  
 Properties: Export-Only Field

5.1.3.1.5.2.1 Lot\_Flow Model

Lot\_Flow -- a submodel of model Flow\_Plan

The lots being consumed from or produced into a Buffer.

The Lot\_Flow model has fields that references these models :  
Lot, Flow\_Plan.

These models have a field that is a Lot\_Flow model :  
Flow\_Plan, Lot.

The key field for this model is lot

lot -- a Lot field of model Lot\_Flow  
The Lot of the buffer\_plan that is being consumed from or produced into.  
Default: None -- this is a key field

quantity -- a Quantity field of model Lot\_Flow  
The Quantity of the lot that is being consumed; this may be less than the total Quantity of the lot, depending upon the Flow's usage\_policy.

The Quantity is negative if this Flow\_Plan is producing into the lot. In that case, this Quantity will be the total Quantity of the lot (but negative).

The Quantity is converted to the unit of the buffer\_plan.buffer.  
Default: 0

consuming\_flow\_plan -- a Flow\_Plan field of model Lot\_Flow  
The Flow\_Plan that consumes the Lot making up this Lot\_Flow.  
Properties: Export-Only Field

upstream\_lot\_flows -- a List[Lot\_Flow] field of model Lot\_Flow  
A list of all Lot\_Flows that feed this one skipping over operations. Recursing with this function can be used for keeping track of Lots and costs for a particular item.  
Properties: command=True Export-Only Field

downstream\_lot\_flows -- a List[Lot\_Flow] field of model Lot\_Flow  
A list of all Lot\_Flows that are fed by this one skipping over operations. Recursing with this function can be used for keeping track of the items that Lot ends up in, and the costs associated with that.

Properties: command=True Export-Only Field

owner -- a Flow\_Plan field of model Lot\_Flow

Properties: Export-Only Field  
5.1.3.1.6 Operation\_State Model

Operation\_State -- a submodel of model Site\_Plan

Operation\_State is an indirect mechanism for reporting the state of Operation\_Plans. It is not normally used interactively -- it is much easier to access the Operation\_Plan directly. The Operation\_State is primarily intended to map state information provided by other systems (ERP, MRP II, SFC, MES, etc.) to the appropriate Operation\_Plan.

The 'state\_spec' extension specifies how the state information itself is expressed. The 'identifier' extension specifies how the Operation\_Plan that this State pertains to is identified. The 'operation\_plans\_list' is all Operation\_Plans that match the criteria. The 'operation\_plan' is the Operation\_Plan that this Operation\_State is applied to. It can be set by the user, and need not match any that appear in 'operation\_plans'.

If zero or more than one Operation\_Plan matches, then 'operation\_plan' will be initially set to [unspecified] and an UNIDENTIFIED\_OP\_STATE Problem will be raised. The 'operation\_plan' must be set interactively to eliminate the Problem.

Operation\_State is usually only interacted with in order to correct errors or to aid or correct the identification of the Operation\_Plan. Important: When an op state is imported into SCP, it will not be attached to an op plan immediately. Currently Operation\_States are attached to Operation\_Plans as a side effect of problem list examination, either directly via a call to Plan.problems, or indirectly via strategy execution. To avoid unexpected behavior, use Plan.problems.count as soon as wip is imported or when the operation\_plan field is set.

The model has selectors:  
identifier, state\_spec.

The Operation\_State model has fields that references these models :  
Operation\_Plan, Site\_Plan.

These models have a field that is a Operation\_State model :  
LINK, UNIDENTIFIED\_OP\_STATE.

The key field for this model is identifier



Models	Operation_State Model
--------	-----------------------

This model may be extended with user-defined fields.

**operation\_plan** - - *a Operation\_Plan field of model Operation\_State*  
The Operation\_Plan that this is reporting about. Once set (or computed) to other than the default [unspecified], the Operation\_State will appear in that Operation\_Plan's 'operation\_states' List and will be used to compute its state.  
Default: [unspecified]

**unattach** - - *a Void field of model Operation\_State*  
Causes the Operation\_State to unattach itself from the currently selected operation\_plan. If the currently selected operation\_plan is unspecified then nothing will happen.  
Properties: command=True Export-Only Field

**operation\_plans** - - *a List(Operation\_Plan) field of model Operation\_State*  
The Operation\_Plans that are identified by the fields associated with the 'identifier' extension.  
Properties: Export-Only Field

**identifier** - - *an extension selector of model Operation\_State*  
Specifies how to identify the Operation\_Plan that this Operation\_State pertains to. For example, it may specify the 'release\_name' of a released Operation\_Plan, or it may specify the 'release\_name' of a released super-Operation and a sub-Operation name, or it may specify a super-Operation name and a sub-Operation name and expect the earliest that fits to be identified, or it may specify an Item name and Lot name, or many other possibilities. Currently, however, the only available Identifier is EARLIEST.  
Default: None -- this is a key field  
Extensions: EARLIEST.

**date** - - *a Date field of model Operation\_State*  
The Date at which this Operation\_State was accurate.  
Default: today

**state\_spec** - - *an extension selector of model Operation\_State*  
Specifies the fields and how they will be used to specify the state of the 'operation\_plan'. For example, a shop floor control system may give cumulative completed quantities, the amount in-process, the percent-complete, the hours completed, the hours remaining, etc. This extension supports easy mapping of such varied input forms into the Operation\_Plan model.  
Default: COMPLETED

Models	Request Model
--------	---------------

Extensions:  
**STARTED, COMPLETED, IN\_FRONT,**

**owner** - - *a Site\_Plan field of model Operation\_State*

Properties: Export-Only Field  
**5.1.3.1.7 Request Model**

**Request** - - *a submodel of model Site\_Plan*

The Request, Delivery\_Request, and Item\_Request models together model requests from one Site to another. The parallel Promise, Delivery\_Promise, and Item\_Promise models together model the supplying Site's commitment back to the requesting Site. These models together implement what is traditionally called "Order Management" or "Demand Management". Simplified variations of those models are often called "orders".

The term "order" is intentionally not used in the model and field names. Order is used to mean very different things in different industries, in different companies in the same industry, and even in different groups in the same company. There are demand orders, sales orders, purchase orders, blanket orders, work orders, maintenance orders, shop orders, and manufacturing orders; many of which are very different from each other. And each of those may mean subtly or drastically different things for different people. To avoid confusing terminology then, we have chosen to avoid the term "order" in model and field names, though we will use it in explanations such as this one.

An Item\_Request can model a typical sales-order line-item. The Delivery\_Request can model an order where all the line items are to be delivered together. A Request can model an order containing groups of line items that are to be delivered at different Dates, or can model a long-term contract or agreement for orders to be placed in the future.

Fundamentally, a Request is a set of Delivery\_Requests that are issued together and a Promise is to be offered for the whole. If the Delivery\_Promises or Item\_Promises are to be offered separately, then they should be placed in separate Requests and Promises.

For long-term negotiated contracts, the 'delivery\_policy' can define how Delivery\_Requests are generated and adjusted. This provides similar functionality to Forecast's generation of forecast Requests and adjustment of those Requests as actual Requests arrive. However, in this case it is to reflect Delivery\_Requests that are part of a larger agreement between two Sites. This is often referred to as a "blanket order" or a "contract".

The supplier's half of the order (e.g., the agreed-to due date) is modeled in the parallel Promise models (Promise, Delivery\_Promise, and Item\_Promise). These are separated from the corresponding Request models for protocol reasons -- only the customer should be modifying the Request, and only the supplier should be modifying the Promise.

In addition to the model separation, there are numerous fields designed to support the typical order protocols that are needed in real situations. The basic flow is as follows: (1) the customer issues a Request, (2) the supplier offers a Promise to that Request which may differ from the Request (may even be zero), and (3a) the customer accepts the offer (adjusting the Request to match the Promise), or (3b) the customer accepts the offer but also queues the Request for more (assuming the Promise was less than Requested), or (3c) the customer deletes the Request. Deadlines may be imposed on these steps. When a Request is issued it may specify a deadline for receiving a Promise. And the Promise may specify a deadline for acceptance (the offer is only good until that deadline).

This protocol is maintained by simple Date fields that keep track of when things happened. By simply comparing the Dates, the state of the protocol is known. The fields are: 'date\_issued' (set by step 1), 'promise.date\_offered' (set by step 2), 'date\_queued' (set by 3b), and 'date\_accepted' (set by 3a and 3b). The deadlines are 'promise\_by' and 'promise.accept\_by' (note that the Request's 'accept\_by' is just the desired deadline).

If 'date\_issued' is after 'promise.date\_offered', then the Request has not been responded to, and a PROMISE\_NOT\_OFFERED Problem will exist (its severity depends upon 'request.promise\_by'). This is resolved by step (2).

If 'promise.date\_offered' is at or after 'date\_issued' and 'date\_accepted' is before 'promise.date\_offered', then the Request has been offered a Promise, but the Promise has not been accepted. A PROMISE\_NOT\_ACCEPTED Problem will exist (its severity depends upon 'promise.accept\_by'). This is resolved by step (3b).

If 'date\_accepted' is at or after 'promise.date\_offered' which is at or after 'date\_issued', then the Promise has been offered and accepted and is now binding on both parties.

If 'date\_queued' is at or after 'promise.date\_offered' which is at or after 'date\_issued', then the Request has been queued for later consideration and a REQUEST\_QUEUED Problem will exist.

The model has selectors:  
delivery\_policy, delivery\_naming.

The Request model has these submodels :  
Delivery\_Request.

The Request model has fields that references these models :  
Delivery\_Request, Promise, Seller\_Plan, Forecast, Site\_Plan.

These models have a field that is a Request model :

Forecast, LINK, Promise, SUPPLIER, Delivery\_Request,  
REQUEST\_NOT\_PLANNED, REQUEST\_PLANNED\_LATE,  
REQUEST\_PLANNED\_EARLY, REQUEST\_PLANNED\_SHORT,  
REQUEST\_PLANNED\_EXCESS, PROMISE\_NOT\_PLANNED,  
PROMISE\_PLANNED\_LATE, PROMISE\_PLANNED\_EARLY,  
PROMISE\_PLANNED\_SHORT, PROMISE\_PLANNED\_EXCESS,  
ACCEPTANCE\_NOT\_PLANNED, ACCEPTANCE\_PLANNED\_LATE,  
ACCEPTANCE\_PLANNED\_EARLY, ACCEPTANCE\_PLANNED\_SHORT,  
REQUEST\_PLANNED\_EXCESS, REQUEST\_PROMISED\_LATE,  
REQUEST\_PROMISED\_EARLY, REQUEST\_PROMISED\_SHORT,  
DELIVERY\_PROMISE\_OVERPRICED, ITEM\_PROMISE\_OVERPRICED,  
PROMISE\_NOT\_CONFIRMED, PROMISE\_NOT\_OFFERED,  
ACCEPTANCE\_INCONSISTENT, REQUEST\_QUEUED,  
DELIVERY\_REQUEST\_NOT\_COORDINATED,  
DELIVERY\_PROMISE\_NOT\_COORDINATED,  
DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED,  
SUPPLY\_PLANNED\_LATE, SUPPLY\_PLANNED\_EARLY,  
SUPPLY\_PLANNED\_SHORT, SUPPLY\_PLANNED\_EXCESS,  
SUPPLY\_PROMISED\_LATE, SUPPLY\_PROMISED\_EARLY,  
SUPPLY\_PROMISED\_SHORT, SUPPLY\_PROMISED\_EXCESS.

The key field for this model is name

This model may be extended with user-defined fields.

name - - a Symbol field of model Request

The name of this Request. The name must be unique among Requests of the 'owner' Site\_Plan.

For single-delivery Requests, this may be the "order ID". The Delivery\_Request name may be the same or may be something simple like "0". For blanket orders or contracts, this is the contract ID.

Often, this 'name' is formed from the customer name and their PO# concatenated together.

Default: None -- this is a key field

**description** - - *a String field of model Request*

A description of or comments about this Request.

Default: none

**delivery\_requests** - - *a list of Delivery\_Request submodels of model Request*

The individual orders that are being requested. Each Delivery\_Request consists of one or more Item\_Requests that are to be delivered together.

**delivery\_policy** - - *an extension selector of model Request*

Defines what 'delivery\_requests' will be created, and what should be created. Defines the forecast orders and how they change as actual orders are introduced.

The default, **FIXED**, specifies that there is no order variation -- the 'delivery\_requests' explicitly list what is being requested. If all the 'delivery\_requests' are 'promised', then (and only then) is this Request 'promised'.

Default: **FIXED**

Extensions:

**FIXED**.

**delivery\_naming** - - *an extension selector of model Request*

Defines how Delivery\_Requests generated by the delivery\_policy should be named.

Default: **NUMBERED**

Extensions:

**NUMBERED**.

**accept\_by** - - *a Date field of model Request*

The Date by which the 'customer\_plan' intends to accept or reject any Promise that is made. This is the Date on which both parties would become committed. And it is the start of the delivery\_lead\_time (which can affect pricing).

Note that this is just the requested 'accept\_by' Date -- the actual deadline for accepting a Promise is in the Promise 'accept\_by'. The supplier considers this requested Date when making the Promise, but may choose a different Date based upon the delivery lead times of its Products. Such differences are part of the negotiation of Request and Promise.

Default: promise\_by + "1 day"

**promise** - - *a Promise field of model Request*

The Promise made in response to this Request. Note that there is always a Promise model for any Request model, but it may not be 'promised' yet. In fact, it could be rejected (Promise of zero). The Promise models either of those answers (or lack of an answer) to this Request.

Properties: Export-Only Field

**promise\_by** - - *a Date field of model Request*

The Date by which an answer to this Request is required. This Request will automatically expire on this Date if no answer is given. The default, oo\_future, indicates that the Request will not expire automatically.

Default: oo\_future

**date\_issued** - - *a Date field of model Request*

The Date that this Request is considered to have been made. If this Date is after the Dates in the 'promise', then this Request is considered to be unanswered. Thus, setting this field to 'now' essentially renews this Request.

Default: 'now'

**date\_accepted** - - *a Date field of model Request*

The Date that the Promise offered for this Request was accepted (agreed to). Until accepted, the 'supplier' need not fill the Promise, and the customer need not accept it. Once offered, it must be accepted prior to the 'accept\_by' Date in the Promise (the 'accept\_by' Date in the Request is just a request).

The Promise has only been accepted when 'date\_accepted' is on or after 'promise.date\_offered', which itself must be on or after 'date\_issued'. So, the default, oo\_past, which is always prior to 'date\_issued', means it has not been accepted.

Default: oo\_past

**date\_queued** - - *a Date field of model Request*

The Date that the 'customer\_plan' asked the 'owner' supplier to queue this Request. There are three possible responses to an inadequate Promise being offered. One is to withdraw (delete) the Request. One is to modify the Request and renew it by setting 'date\_issued' to after 'promise.date\_offered'. One is to queue the Request, by setting this field, 'date\_queued', to on or after 'promise.date\_offered'.

A queued Request is one that will be reconsidered if conditions at the supplier change such that new Requests can be satisfied. When that occurs, the 'owner' supplier will immediately consider any queued Requests. In that way, the 'customer\_plan' need not keep renewing a rejected Request just in case capacity has freed up.

Default: oo\_past

<i>Models</i>	<i>Request Model</i>
---------------	----------------------

**cancel** - - *a Void field of model Request*

This command cancels all the delivery\_requests and item\_requests associated with this request. All the item\_requests will be marked as cancelled. None of the item\_requests or delivery\_requests will be deleted. The associated delivery and receiving plans will be destroyed.

Properties:    command=True    Export-Only Field

**plan\_to\_satisfy** - - *a Void field of model Request*

This command creates plans to satisfy this Request. It does this by simply calling 'plan\_to\_satisfy' on each of its 'delivery\_requests'. Thus, it is equivalent to 'delivery\_requests.for\_each(#.plan\_to\_satisfy)';

Alternatively, 'promise.plan\_to\_satisfy' can be performed to call 'plan\_to\_satisfy' on each of the 'delivery\_promises'.

Properties:    command=True    Export-Only Field

**plan\_as\_requested** - - *a Void field of model Request*

This command sets the plan for this Request such that it is attempting to satisfy this Request, and all of its 'delivery\_requests' and 'item\_requests'.

Properties:    command=True    Export-Only Field

**seller\_plan** - - *a Seller\_Plan field of model Request*

The Seller\_Plan of the Seller that is responsible for this agreement. All

'delivery\_requests' of this Request will have this or a member as its 'seller'. For example, a blanket Request could be established through an Seller organization, but it is the 'members' that eventually make the individual Delivery\_Requests.

Default:    [unspecified]

**forecast** - - *a Forecast field of model Request*

This is the Forecast that generated this Request. If nonexistent, then this Request is an actual (customer) Request.

Properties:    Export-Only Field

**customer\_plan** - - *a Site\_Plan field of model Request*

The Site\_Plan of the Site placing this Request (or forecasted to be placing this Request) on the 'owner' Site\_Plan. All 'delivery\_requests' of this Request will have this or a member Site as its 'customer\_plan'. For example, a blanket Request could be established by an ORGANIZATION Site, but it is the members that eventually make the individual Delivery\_Requests.

Default:    [unspecified]

<i>Models</i>	<i>Request Model</i>
---------------	----------------------

**name\_from\_customer** - - *a Symbol field of model Request*

The customer's name for this Request. This may be the same as 'name' or different. For example, this is often the customer's PO# (purchase order number). However, the supplier will often name Requests differently -- after all, there is no guarantee that these names (PO#'s) will be unique, since they come from different customers.

Default:    'name'

**delivery\_name** - - *a String field of model Request*

The name of the customer to which this Request should be delivered.

Default:    customer.site.delivery\_name

**delivery\_contact** - - *a String field of model Request*

The name of the individual to whom this Request should be delivered.

Default:    customer.site.delivery\_contact

**delivery\_phone** - - *a String field of model Request*

The phone number of the 'delivery\_contact' to which this Request should be delivered.

Default:    customer.site.delivery\_phone

**delivery\_fax** - - *a String field of model Request*

The fax phone number of the 'delivery\_contact' to which this Request should be delivered.

Default:    customer.site.delivery\_fax

**delivery\_address** - - *a String field of model Request*

The street address to which this Request should be delivered.

Default:    customer.site.delivery\_address

**delivery\_city** - - *a String field of model Request*

The city (township) to which this Request should be delivered.

Default:    customer.site.delivery\_city

**delivery\_state** - - *a String field of model Request*

The state (province, territory) to which this Request should be delivered.

Default:    customer.site.delivery\_state

**delivery\_country** - - *a String field of model Request*

The country (nation) to which this Request should be delivered.

Default:    customer.site.delivery\_country

Models	Request Model
--------	---------------

delivery\_postal\_code - - *a String field of model Request*  
 The postal\_code (zip code) to which this Request should be delivered.  
 Default: customer.site.delivery\_postal\_code

last\_change - - *a Date field of model Request*  
 The Date at which the most recent change was made to this Request. This is set by setting this or any other field in this model or the 'delivery\_requests'. Note that setting this directly will set it to the specified Date, which may not be the current Date.

This is used to support the 'changed\_requests' functions of Site, which provide net-change Exporting of Requests.  
 Default: now

issued - - *a Date field of model Request*  
 Obsolete! This field has been renamed 'date\_issued' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.  
 Default: 'now'  
 Properties: obsolete=True

accepted - - *a Date field of model Request*  
 Obsolete! This field has been renamed 'date\_accepted' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.  
 Default: oo\_past  
 Properties: obsolete=True

queued - - *a Date field of model Request*  
 Obsolete! This field has been renamed 'date\_queued' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.  
 Default: oo\_past  
 Properties: obsolete=True

problems, problems (Date\_Range) , problems (Problem\_Category) , problems (Date\_Range, Problem\_Category) - - *a List(Problem) field of model Request*  
 The Problems associated with this Request, including any problems that are associated with the underlying Delivery\_Request and Item\_Request models. If passed a Date\_Range, only the Problems whose 'dates' overlap are returned. If passed a Problem\_Category, only the Problems with that 'category' are returned.  
 Properties: Export-Only Field

Models	Request Model
--------	---------------

owner - - *a Site\_Plan field of model Request*  
 The Site\_Plan upon which this Request is placed -- this Site is responsible for promising and filling the 'orders'.  
 Properties: Export-Only Field

Models	Delivery_Request Model
--------	------------------------

### 5.1.3.1.7.1 Delivery\_Request Model

#### Delivery\_Request --- a submodel of model Request

Delivery\_Request models a request for a set of Items to be delivered together. Several Delivery\_Requests may make up the 'owner' Request. The Delivery\_Request specifies the 'due' Date or Date\_Range. The Item\_Requests within the Delivery\_Request specify the Items and quantities.

The model has selectors:  
promising\_policy, fulfillment\_policy.

The Delivery\_Request model has these submodels :  
Item\_Request.

The Delivery\_Request model has fields that references these models :  
Forecast\_Entry, Delivery\_Promise, Item\_Request, Seller\_Plan, Site\_Plan, Problem.

These models have a field that is a Delivery\_Request model :

Request, Delivery\_Promise, Item\_Request, REQUEST\_NOT\_PLANNED, REQUEST\_PLANNED\_LATE, REQUEST\_PLANNED\_EARLY, REQUEST\_PLANNED\_SHORT, REQUEST\_PLANNED\_EXCESS, PROMISE\_NOT\_PLANNED, PROMISE\_PLANNED\_LATE, PROMISE\_PLANNED\_EARLY, PROMISE\_PLANNED\_SHORT, PROMISE\_PLANNED\_EXCESS, ACCEPTANCE\_NOT\_PLANNED, ACCEPTANCE\_PLANNED\_LATE, ACCEPTANCE\_PLANNED\_EARLY, ACCEPTANCE\_PLANNED\_SHORT, ACCEPTANCE\_PLANNED\_EXCESS, REQUEST\_PROMISED\_LATE, REQUEST\_PROMISED\_EARLY, REQUEST\_PROMISED\_SHORT, REQUEST\_PROMISED\_EXCESS, ITEM\_PROMISE\_OVERPRICED, DELIVERY\_PROMISE\_OVERPRICED, PROMISE\_NOT\_CONFIRMED, DELIVERY\_REQUEST\_NOT\_COORDINATED, DELIVERY\_PROMISE\_NOT\_COORDINATED, DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED, SUPPLY\_PLANNED\_LATE, SUPPLY\_PLANNED\_EARLY, SUPPLY\_PLANNED\_SHORT, SUPPLY\_PLANNED\_EXCESS, SUPPLY\_PROMISED\_LATE, SUPPLY\_PROMISED\_EARLY, SUPPLY\_PROMISED\_SHORT, SUPPLY\_PROMISED\_EXCESS.

The key field for this model is name  
This model may be extended with user-defined fields.

Models	Delivery_Request Model
--------	------------------------

name --- a Symbol field of model Delivery\_Request

The name of this Delivery\_Request. In the common case of a single Delivery\_Request in the Request, this is typically the same as the 'owner.name', the order-id. Alternatively, it could be something simple like "0". In the case of contracts, this is typically sub-order-id or delivery-id or billing-id. In the case of a Forecast\_Request or generated orders, this 'name' is typically generated as well.

Default: None -- this is a key field

actual --- a Logical field of model Delivery\_Request

If "true", then this is an actual (customer) Request. Otherwise, this Delivery\_Request was generated either by the 'delivery\_policy' of the 'owner' Request, or the 'owner' Request itself was generated by its 'forecast\_entry'.

Properties: Export-Only Field

forecast\_entry --- a Forecast\_Entry field of model Delivery\_Request

If nonexistent, then this is an actual (customer) Delivery\_Request. Otherwise, this is the Forecast\_Entry that generated this Delivery\_Request.

Properties: Export-Only Field

delivery\_promise --- a Delivery\_Promise field of model Delivery\_Request

The Delivery\_Promise that has been made in response to this Delivery\_Request.

Properties: Export-Only Field

item\_requests --- a list of Item\_Request submodels of model Delivery\_Request

The individual order-line-items (Items) being requested.

due --- a Date\_Range field of model Delivery\_Request

The Date\_Range within which the delivery should be made to the 'customer.plan'. Sometime between 'due.start' and 'due.end', the full Quantity specified in each of the 'item\_requests' should be satisfied.

If the 'due.end' Date is beyond the Plan horizon, (including the default "oo\_future"), then this is considered to be a request for nothing. No Problem or Field\_Error is created. In effect, this Delivery\_Request is "turned off". This state is particularly useful while importing a set of requests.

Default: forever

promising\_policy --- an extension selector of model Delivery\_Request

This policy specifies the constraints on offering a Promise for the corresponding Request.

Default: ASAP

Extensions:

BUCKETED\_ALL, BUCKETED\_ASAP, ON\_TIME, ALL, ALL\_ON\_TIME, ASAP, ASAP\_MONTHLY, BUCKETED\_ALLOCATION, BUCKETED\_ALL\_MIN\_PRICE, BUCKETED\_MIN\_PRICE\_ASAP, SHIP\_IN\_RATIO.

fulfillment\_policy - - *an extension selector of model Delivery\_Request*

This policy specifies restrictions on how a delivery is fulfilled.

Default: UNRESTRICTED

Extensions:

ON\_TIME, FULL\_QUANTITIES\_OF\_ALL\_ITEMS, UNRESTRICTED.

max\_price - - *a Money field of model Delivery\_Request*

The maximum price desired by the customer\_plan for this Delivery\_Request. This is used as a guideline (not hard constraint) in order quoting.

Default: oo

seller\_plan - - *a Seller\_Plan field of model Delivery\_Request*

The Seller that is responsible for this Delivery\_Request. This may be the same as or a member of owner:seller\_plan.

Default: owner:seller\_plan

customer\_plan - - *a Site\_Plan field of model Delivery\_Request*

The Site placing this Delivery\_Request. This may be the same as or a member of owner:customer\_plan.

Default: owner:customer\_plan

name\_from\_customer - - *a Symbol field of model Delivery\_Request*

The customer's name for this Delivery\_Request. This may be the same as name' or different. For example, this is often the customer's PO# (purchase order number). However, the supplier will often name Requests differently -- after all, there is no guarantee that names (PO#s) will be unique, since they come from different customers.

Default: name'

rank - - *a Number field of model Delivery\_Request*

The customer-specified rank, weighting, importance of this Delivery\_Request relative to all other Delivery\_Requests from this customer.

Note, as with all rank's, the higher the Number, the higher the rank, the greater the importance, the larger the weight.

Default: 0

delivery\_name - - *a String field of model Delivery\_Request*

The name of the customer to which this Delivery\_Request should be delivered.

Default: owner:delivery\_name

delivery\_contact - - *a String field of model Delivery\_Request*

The name of the individual to whom this Delivery\_Request should be delivered.

Default: owner:delivery\_contact

delivery\_phone - - *a String field of model Delivery\_Request*

The phone number of the delivery\_contact to which this Delivery\_Request should be delivered.

Default: owner:delivery\_phone

delivery\_fax - - *a String field of model Delivery\_Request*

The fax phone number of the delivery\_contact to which this Delivery\_Request should be delivered.

Default: owner:delivery\_fax

delivery\_address - - *a String field of model Delivery\_Request*

The street address to which this Delivery\_Request should be delivered.

Default: owner:delivery\_address

delivery\_city - - *a String field of model Delivery\_Request*

The city (township) to which this Delivery\_Request should be delivered.

Default: owner:delivery\_city

delivery\_state - - *a String field of model Delivery\_Request*

The state (province, territory) to which this Delivery\_Request should be delivered.

Default: owner:delivery\_state

delivery\_country - - *a String field of model Delivery\_Request*

The country (nation) to which this Delivery\_Request should be delivered.

Default: owner:delivery\_country

delivery\_postal\_code - - *a String field of model Delivery\_Request*

The postal code (zip code) to which this Delivery\_Request should be delivered.

Default: owner:delivery\_postal\_code

**promised\_late** - - *a Time field of model Delivery\_Request*  
If this Delivery\_Request promises to deliver later than requested by 'item\_request', then this returns how much later; otherwise it returns "0". This is infinite if 'unanswered' is "true". If this is non-zero and finite, then there is either a REQUEST\_PROMISED\_LATE Problem for this Delivery\_Request. If infinite, then there is a DELIVERY\_REQUEST\_UNANSWERED Problem.  
Properties:    Export-Only Field

**promised\_early** - - *a Time field of model Delivery\_Request*  
If this item Promise promises to deliver earlier than requested by 'item\_request', then this returns how much earlier; otherwise it returns "0". This is "0" if 'unanswered' is "true". If this is non-zero, then there is a REQUEST\_PROMISED\_EARLY Problem for this item Promise.  
Properties:    Export-Only Field

**promised\_overpriced** - - *a Money field of model Delivery\_Request*  
If this Delivery\_Request offers a price that is more than the 'delivery\_request\_max\_price', then this returns how much more; otherwise it returns "0". This is "0" if 'unanswered' is "true". If this is non-zero, then there is a REQUEST\_PROMISED\_OVERPRICED Problem for this item Promise.  
Properties:    Export-Only Field

**cancel** - - *a Void field of model Delivery\_Request*  
This command cancels all the item\_requests associated with this delivery\_request. All the item\_requests will be marked as cancelled but none of them will be deleted. All the associated delivery and receiving plans will be destroyed.  
Properties:    command=True    Export-Only Field

**plan\_to\_satisfy** - - *a Void field of model Delivery\_Request*  
This command sets the plan for this Delivery\_Request such that it is attempting to satisfy this Delivery\_Request, and all of its 'item\_requests'. The 'delivery\_promise.delivery\_plan\_motive' will be set to match this Delivery\_Request's 'due' and all its 'item\_requests' quantity's.

Alternatively, 'delivery\_promise.plan\_to\_satisfy' can be performed to set the plan to attempt to satisfy the promises for this request. Or, the 'item\_promise.delivery\_plan\_motive' can be set directly to attempt any subset of this Delivery\_Request or its 'delivery\_promise'.  
Properties:    command=True    Export-Only Field

**plan\_as\_requested** - - *a Void field of model Delivery\_Request*  
This command sets the plan for this Delivery\_Request such that it is attempting to satisfy this Delivery\_Request, and all of its 'item\_requests'.  
Properties:    command=True    Export-Only Field

**not\_planned** - - *a Logical field of model Delivery\_Request*  
If "true", then no plan has been formed to meet at least parts of this Delivery\_Request. This is "true" if 'due' is finite ('unanswered' is "false") and at least one of the 'item\_requests' is 'unplanned'. If "true", then there is a REQUEST\_NOT\_PLANNED Problem for those unplanned item\_Promises.  
Properties:    Export-Only Field

**planned\_late** - - *a Time field of model Delivery\_Request*  
Returns how late this Delivery\_Request is currently planned to be fully delivered. It is the latest of each of the 'item\_requests'. Note that 'delivery\_request\_planned\_late' is equivalent to 'max(delivery\_request.item\_requests.for\_each(#planned\_late))'. If non-zero, then there is a REQUEST\_PLANNED\_LATE Problem for at least one of the 'item\_requests'.

Unlike the individual 'item\_requests', a Delivery\_Request can have both 'planned\_late' and 'planned\_early' non-zero.  
Properties:    Export-Only Field

**planned\_early** - - *a Logical field of model Delivery\_Request*  
Returns how early this Delivery\_Request is currently planned to be delivered, at least in part. It is the earliest of each of the 'item\_requests'. Note that 'delivery\_request\_planned\_early' is equivalent to 'min(delivery\_request.item\_requests.for\_each(#planned\_early))'. If non-zero, then there is a REQUEST\_PLANNED\_EARLY Problem for at least one of the 'item\_requests'.

Unlike the individual 'item\_requests', a Delivery\_Request can have both 'planned\_late' and 'planned\_early' non-zero.  
Properties:    Export-Only Field

**last\_change** - - *a Date field of model Delivery\_Request*  
The Date at which the most recent change was made to this Delivery\_Request. This is set by setting this or any other field in this model or the 'item\_requests'. Note that setting this directly will set it to the specified Date, which may not be the current Date.



<i>Models</i>	<i>Delivery_Request Model</i>
---------------	-------------------------------

This is used to support the 'changed\_requests' functions of Site, which provide net-change Exporting of Requests.  
 Default:   now

problems, problems (Date\_Range) , problems (Problem\_Category) , problems (Date\_Range, Problem\_Category)   -- a List(Problem) *field of model* Delivery\_Request  
 The Problems associated with this Delivery\_Request, including any problems that are associated with the underlying Item\_Request model. If passed a Date\_Range, only the Problems whose 'dates' overlap are returned. If passed a Problem\_Category, only the Problems with that 'category' are returned.  
 Properties:   Export-Only Field

promised\_date\_problem   -- a Problem *field of model* Delivery\_Request  
 Obsolete! Use Delivery\_Request problems'. The REQUEST\_PROMISED\_DATE Problem, if any, that exists between this Item\_Promise and the Item\_request' it is answering. See fields 'unanswered', 'promised\_late', and 'promised\_early' for alternate ways to test for REQUEST\_UNANSWERED, REQUEST\_PROMISED\_LATE, and REQUEST\_PROMISED\_EARLY Problems, respectively.  
 Properties:   obsolete=True   Export-Only Field

promised\_price\_problem   -- a Problem *field of model* Delivery\_Request  
 Obsolete! Use Delivery\_Request problems'. The REQUEST\_PROMISED\_OVERPRICED Problem, if any, that exists between this Item\_Promise and the Item\_request' it is answering. See the field 'overpriced' for an alternate way to test for this Problem.  
 Properties:   obsolete=True   Export-Only Field

plan\_problems   -- a List(Problem) *field of model* Delivery\_Request  
 Obsolete! Use Delivery\_Request problems'. A List of the REQUEST\_PLAN Problems of the Item\_requests', if any. See fields 'unplanned', 'planned\_late', and 'planned\_early' for alternate ways to test for REQUEST\_NOT\_PLANNED, REQUEST\_PLANNED\_LATE, and REQUEST\_PLANNED\_EARLY Problems, respectively.

Note well: the lack of a Problem here only indicates that a plan to meet this Request is being attempted. There may be feasibility Problems with the plans associated with fulfilling this Request in addition to any Problem here.  
 Properties:   obsolete=True   Export-Only Field

<i>Models</i>	<i>Delivery_Request Model</i>
---------------	-------------------------------

owner   -- a Request *field of model* Delivery\_Request  
 Properties:   Export-Only Field

### 5.1.3.1.7.1.1 Item\_Request Model

#### Item\_Request -- a submodel of model Delivery\_Request

A Item\_Request is a request for supply of a Quantity of a particular Item. This generally corresponds to the order "line item". Note that the due Date is specified by the 'owner' Delivery\_Request, which allows several line items to be defined to be delivered together (same due Date\_Range plus they should all be late if one is).

The Item is specified by the 'configuration' field, which contains an 'Item' field. The Configuration field supports configure-to-order products, and other non-STANDARD Items.

The Item\_Request model has fields that references these models :  
Configuration, Item, Item\_Promise, Operation\_Plan, Problem, Delivery\_Request.

These models have a field that is a Item\_Request model :  
Delivery\_Request, Item\_Promise, REQUEST\_NOT\_PLANNED, REQUEST\_PLANNED\_LATE, REQUEST\_PLANNED\_EARLY, REQUEST\_PLANNED\_SHORT, REQUEST\_PLANNED\_EXCESS, PROMISE\_NOT\_PLANNED, PROMISE\_PLANNED\_LATE, PROMISE\_PLANNED\_EARLY, PROMISE\_PLANNED\_SHORT, PROMISE\_PLANNED\_EXCESS, ACCEPTANCE\_NOT\_PLANNED, ACCEPTANCE\_PLANNED\_LATE, ACCEPTANCE\_PLANNED\_EARLY, ACCEPTANCE\_PLANNED\_SHORT, ACCEPTANCE\_PLANNED\_EXCESS, DELIVER, REQUEST\_PROMISED\_SHORT, REQUEST\_PROMISED\_EXCESS, ITEM\_PROMISE\_OVERPRICED, SUPPLY\_PLANNED\_LATE, SUPPLY\_PLANNED\_EARLY, SUPPLY\_PLANNED\_SHORT, SUPPLY\_PLANNED\_EXCESS, SUPPLY\_PROMISED\_LATE, SUPPLY\_PROMISED\_EARLY, SUPPLY\_PROMISED\_SHORT, SUPPLY\_PROMISED\_EXCESS, REQUEST\_FIXED, REQUEST\_FIXED\_WITH\_ANALYSIS.

The key field for this model is name  
This model may be extended with user-defined fields.

name -- a Symbol field of model Item\_Request  
The name of this Item\_Request. Depending upon the environment, this may be the order line-item number, it may be the name of the Item (Item family, or pseudo item), or otherwise. In the case of single Item orders, it may be the order-id or an empty String. It simply must be unique within the Delivery\_Request, but otherwise can be whatever is meaningful.

Default: None -- this is a key field

#### configuration -- a Configuration field of model Item\_Request

The Item being requested and the specific characteristics desired. The Item may be a STANDARD Item (no characteristics to specify), a Request-specific configuration of a OPTIONED Item, or a Request-specific CUSTOM Item (among others).

This is not settable itself, but you can set the fields of this Configuration.

If the 'Item' is [unspecified] (the default), then this is considered to be a request for nothing. No Problem or Field\_Error is created. In effect, this Item\_Request is "turned off". Similarly, setting 'quantity' to zero (the default) or 'owner.due' to "oo\_future" (the default) will effectively "turn off" this Item\_Request. This state is particularly useful while importing a set of requests.

Properties: command=True Export-Only Field

#### Item -- a Item field of model Item\_Request

This is simply a shortcut to configuration.item.

If the 'Item' is [unspecified] (the default), then this is considered to be a request for nothing. No Problem or Field\_Error is created. In effect, this Item\_Request is "turned off". Similarly, setting 'quantity' to zero (the default) or 'owner.due' to "oo\_future" (the default) will effectively "turn off" this Item\_Request. This state is particularly useful while importing a set of requests.

Default: [unspecified]

#### quantity -- a Quantity\_Range field of model Item\_Request

The range of Quantity of the 'configuration' that is being requested as part of the 'owner' Delivery\_Request.

Often this is a zero-interval range -- a particular Quantity is being requested. But in some industries, particular where the yield varies or the Items are specialized, giving a range from the minimum you need to the maximum that you will accept can prevent unnecessary waste, making the supply chain more economical.

These Quantity's are converted to the 'unit' of the 'Item' being requested.

Models	Item_Request Model
--------	--------------------

If the 'quantity' is "0" (the default), then this is considered to be a request for nothing. No Problem or Field\_Error is created. In effect, this Item\_Request is "turned off". Similarly, setting 'item' to (unspecified) (the default) or 'owner.due' to "oo\_future" (the default) will effectively "turn off" this Item\_Request. This state is particularly useful while importing a set of requests.  
Default: 0

**cancel** - - *a Void field of model Item\_Request*  
This command marks this Item\_Request as cancelled but does not destroy it. This command will destroy the associated delivery and receiving plans.  
Properties: command=True Export-Only Field

**cancelled** - - *a Logical field of model Item\_Request*  
If "true", then this Item\_Request has been cancelled. Otherwise, this Item\_Request is active and should be considered when promising and planning.  
Default: false

**item\_promise** - - *a Item\_Promise field of model Item\_Request*  
The Item\_Promise that has been made in response to this Item\_Request.  
Properties: Export-Only Field

**max\_price** - - *a Money field of model Item\_Request*  
The maximum price desired by the 'customer' for this Item\_Request. This is used as a guideline (not hard limit) in order quoting.  
Default: oo

**promised\_short** - - *a Quantity field of model Item\_Request*  
If this 'item\_promise' promises less than requested by this Item\_Request, then this returns how much less; otherwise it returns "0". This is "0" if 'quantity' is "0" or 'owner.due' is infinite. If this is non-zero, then there is a REQUEST\_PROMISED\_SHORT Problem for this Item\_Request.  
Properties: Export-Only Field

**promised\_excess** - - *a Quantity field of model Item\_Request*  
If this 'item\_promise' promises more than requested by this Item\_Request, then this returns how much more; otherwise it returns "0". This is "0" if 'quantity' is "0" or 'owner.due' is infinite. If this is non-zero, then there is a REQUEST\_PROMISED\_EXCESS Problem for this Item\_Request.  
Properties: Export-Only Field

Models	Item_Request Model
--------	--------------------

**promised\_overpriced** - - *a Money field of model Item\_Request*  
If this 'item\_promise' offers a 'price' that is more than the 'max\_price', then this returns how much more; otherwise it returns "0". This is "0" if 'quantity' is "0" or 'owner.due' is infinite. If this is non-zero, then there is a REQUEST\_PROMISED\_OVERPRICED Problem for this Item\_Request.  
Properties: Export-Only Field

**delivery\_plan** - - *a Operation\_Plan field of model Item\_Request*  
The Operation that has been planned to deliver this Item\_Request. It is the same as 'item\_promise.delivery\_plan'.  
Properties: Export-Only Field

**plan\_to\_satisfy** - - *a Void field of model Item\_Request*  
This command sets the plan for this Item\_Request such that it is attempting to satisfy this Request. The 'item\_promise.delivery\_plan.motive' will be set to match this Item\_Request's 'quantity' and its 'delivery\_request's 'due' Dates.

Alternatively, 'item\_promise.plan\_to\_satisfy' can be performed to set the plan to attempt to satisfy the promises for this request. Or, the 'item\_promise.delivery\_plan.motive' can be set directly to attempt any subset of this Item\_Request or its 'item\_promise'.  
Properties: command=True Export-Only Field

**receiving\_plan** - - *a Operation\_Plan field of model Item\_Request*  
The Operation that has been planned to receive this Item\_Request.  
Properties: Export-Only Field

**plan\_as\_requested** - - *a Void field of model Item\_Request*  
This command creates or re-sizes the receiving\_plan for this Item\_Request such that it satisfy this Request.  
Properties: command=True Export-Only Field

**not\_planned** - - *a Logical field of model Item\_Request*  
If "true", then no Operation has been planned to satisfy this Item\_Request. This is "false" if 'owner.unanswered' is "true". If "true", then there is a REQUEST\_NOT\_PLANNED Problem for this Item\_Request.  
Properties: Export-Only Field

**planned\_late** - - *a Time field of model Item\_Request*

If the plan for this Item\_Request is planned for later than requested, then this returns how much later; otherwise it returns "0". This is "0" if 'quantity' is "0" or 'owner.due' is infinite. If this is non-zero, then there is a REQUEST\_PLANNED\_LATE Problem for this Item\_Request.

Properties: Export-Only Field

**planned\_early** - - *a Time field of model Item\_Request*

If the plan for this Item\_Request is planned for earlier than requested, then this returns how much earlier; otherwise it returns "0". This is "0" if 'quantity' is "0" or 'owner.due' is infinite. If this is non-zero, then there is a REQUEST\_PLANNED\_EARLY Problem for this Item\_Request.

Properties: Export-Only Field

**planned\_short** - - *a Quantity field of model Item\_Request*

If the plan for this Item\_Request is planned to deliver less than requested, then this returns how much less; otherwise it returns "0". This is "0" if 'quantity' is "0" or 'owner.due' is infinite. If this is non-zero, then there is a REQUEST\_PLANNED\_SHORT Problem for this Item\_Request.

Properties: Export-Only Field

**planned\_excess** - - *a Quantity field of model Item\_Request*

If the plan for this Item\_Request is planned to deliver more than requested, then this returns how much more; otherwise it returns "0". This is "0" if 'quantity' is "0" or 'owner.due' is infinite. If this is non-zero, then there is a REQUEST\_PLANNED\_EXCESS Problem for this Item\_Request.

Properties: Export-Only Field

**last\_change** - - *a Date field of model Item\_Request*

The Date at which the most recent change was made to this Item\_Request. This is set by setting this or any other field in this model. Note that setting this directly will set it to the specified Date, which may not be the current Date.

This is used to support the 'changed\_requests' functions of Site, which provide net-change Exporting of Requests.

This is the same as calling owner.r\_last\_change.

Default: now

**problems**, **problems (Date\_Range)** , **problems (Problem\_Category)** , **problems (Date\_Range, Problem\_Category)** - - *a List(Problem) field of model Item\_Request*

The Problems associated with this Item\_Request. If passed a Date\_Range, only the Problems whose 'dates' overlap are returned. If passed a Problem\_Category, only the Problems with that 'category' are returned.

Properties: Export-Only Field

**promised\_quantity\_problem** - - *a Problem field of model Item\_Request*

Obsolete! Use Item\_Request 'problems'. The REQUEST\_PROMISED\_QUANTITY Problem, if any, that exists between this Item\_Request and the 'item\_promise' answering it. See fields 'promised\_short' and 'promised\_excess' for more convenient ways to test for REQUEST\_PROMISED\_SHORT and REQUEST\_PROMISED\_EXCESS Problems, respectively. This will be nonexistent if there is no such Problem or if 'owner.unanswered' is "true".

Properties: obsolete=True Export-Only Field

**promised\_price\_problem** - - *a Problem field of model Item\_Request*

Obsolete! Use Item\_Request 'problems'. The REQUEST\_PROMISED\_OVERPRICED Problem, if any, that exists between this Item\_Request and the 'item\_promise' answering it. See the field 'overpriced' for an alternate way to test for this Problem. This will be nonexistent if there is no such Problem or if 'owner.unanswered' is "true".

Properties: obsolete=True Export-Only Field

**planned\_date\_problem** - - *a Problem field of model Item\_Request*

Obsolete! Use Item\_Request 'problems'. The REQUEST\_PLANNED\_DATE Problem, if any, that exists between this Item\_Request and the Operation\_Plan currently planned to satisfy it ('delivery\_plan'). See fields 'unplanned', 'planned\_late', and 'planned\_early' for alternate ways to test for REQUEST\_NOT\_PLANNED, REQUEST\_PLANNED\_LATE, and REQUEST\_PLANNED\_EARLY Problems, respectively.

Note well: the lack of a Problem here only indicates that a plan to meet this Request is being attempted. There may be feasibility Problems with the plans associated with fulfilling this Promise in addition to any Problem here.

Properties: obsolete=True Export-Only Field

`planned_quantity_problem` -- a Problem field of model Item\_Request  
Obsolete! Use Item\_Request 'problems'. The REQUEST\_PLANNED\_QUANTITY Problem, if any, that exists between this Item\_Promise and the Operation\_Plan currently planned to satisfy it (`delivery_plan`). See fields 'unplanned', 'planned\_short', and 'planned\_excess' for more convenient ways to test for REQUEST\_NOT\_PLANNED, REQUEST\_PLANNED\_SHORT, and REQUEST\_PLANNED\_EXCESS Problems, respectively.

Note well: the lack of a Problem here only indicates that a plan to meet this Request is being attempted. There may be feasibility Problems with the plans associated with fulfilling this Promise in addition to any Problem here.

Properties: `obsolete=True` Export-Only Field

owner -- a Delivery\_Request field of model Item\_Request

Properties: Export-Only Field

### 5.1.3.1.8 Promise Model

Promise -- a submodel of model Site\_Plan

A Promise models the 'owner' supplier's response to a 'customer's Request. Note that a Promise model exists for every Request model. However, the Promise may not be 'offered' yet; or may be offered, but offering zero Quantity or to be delivered by the infinite future (Request has been rejected). The Promise models either of those responses, or the lack of an offer, to the Request.

Once offered, a Promise models a commitment from the 'owner' to supply a set of deliveries, each with a set of Items. It also becomes, once accepted, a commitment from the 'customer' to purchase the supplied Items.

A Promise can be offered to actual Requests from customers, or to forecasted Requests. In either case, the commitment from the supplier is real. Promises to forecast Requests can be 'consumed' in order to make Promises for actual Requests. Such consumption does not require the involvement of the supplier. Thus, Promises made by the 'supplier' for forecast Requests that have not yet been consumed by actual Requests represent ATP (Available-To-Promise). As such, Promises for forecast Requests are as binding as Promises for actual Requests.

For a Promise to be offered for a Request, a Plan may be created that can satisfy each of the Item\_Requests within each of the Delivery\_Requests of the Request. If no Plan can be found that fully satisfies the Request, a lesser Promise may still be offered. In that case, the Promise should detail what is being offered corresponding to each element of the Request. For each Item\_Request is needed an Item\_Promise, and for each Delivery\_Request a Delivery\_Promise. Thus, the Promise models form a parallel hierarchy to the Request models they are promising.

For the case of Requests that model contracts or "blanket" orders, the exact Delivery\_Requests will not have been generated yet. Rather, the Promise commits to delivering any Delivery\_Requests that can be generated within the bounds specified by the 'delivery\_policy'. To model and plan for that, the 'delivery\_policy' will generate forecasted Delivery\_Requests that mimic the actual Delivery\_Requests that are allowed. Plans will be made to satisfy those 'estimated' Delivery\_Requests, and the Promise will be based upon those Plans.

For more detail on Requests, the hierarchy, or the normal Request-Promise protocol, please see the Request model documentation.

The model has selectors:  
`delivery_policy`.

The Promise model has these submodels:  
Delivery\_Promise.

The Promise model has fields that references these models:  
Request, Delivery\_Promise, Acceptance, Site\_Plan.

These models have a field that is a Promise model:

LINK, Request, Acceptance, SUPPLIER, Delivery\_Promise, REQUEST\_NOT\_PLANNED, REQUEST\_PLANNED\_LATE, REQUEST\_PLANNED\_EARLY, REQUEST\_PLANNED\_SHORT, REQUEST\_PLANNED\_EXCESS, PROMISE\_NOT\_PLANNED, PROMISE\_PLANNED\_LATE, PROMISE\_PLANNED\_EARLY, PROMISE\_PLANNED\_SHORT, PROMISE\_PLANNED\_EXCESS, ACCEPTANCE\_NOT\_PLANNED, ACCEPTANCE\_PLANNED\_LATE, ACCEPTANCE\_PLANNED\_EARLY, ACCEPTANCE\_PLANNED\_SHORT, REQUEST\_PROMISED\_EARLY, REQUEST\_PROMISED\_SHORT, REQUEST\_PROMISED\_EXCESS, ITEM\_PROMISE\_OVERPRICED, DELIVERY\_PROMISE\_OVERPRICED, PROMISE\_NOT\_OFFERED, PROMISE\_NOT\_CONFIRMED, PROMISE\_NOT\_ACCEPTED,

ACCEPTANCE\_INCONSISTENT, REQUEST\_QUEUED,  
DELIVERY\_REQUEST\_NOT\_COORDINATED,  
DELIVERY\_PROMISE\_NOT\_COORDINATED,  
DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED,  
SUPPLY\_PLANNED\_LATE, SUPPLY\_PLANNED\_EARLY,  
SUPPLY\_PLANNED\_SHORT, SUPPLY\_PLANNED\_EXCESS,  
SUPPLY\_PROMISED\_LATE, SUPPLY\_PROMISED\_EARLY,  
SUPPLY\_PROMISED\_SHORT, SUPPLY\_PROMISED\_EXCESS.

The key field for this model is request  
This model may be extended with user-defined fields.

request - - *a Request field of model Promise*  
The Request to which this Promise responds.  
Default: None -- this is a key field  
Properties: Export-Only Field

delivery\_promises - - *a list of Delivery\_Promise submodels of model Promise*  
The individual promises for delivery\_requests.

delivery\_policy - - *an extension selector of model Promise*  
Defines how 'delivery\_promises' may be adjusted, without "changing" the promise.  
This is typically the same as 'request.delivery\_policy', but need not be. The 'owner'  
may promise a different policy than the 'customer' requested. Of course, the 'customer'  
need not accept the Promise.  
Default: FIXED  
Extensions: FIXED  
Properties: Export-Only Field

acceptance - - *a Acceptance field of model Promise*  
The acceptance in response to this promise  
Properties: Export-Only Field

accept\_by - - *a Date field of model Promise*  
Once offered, this Promise is just an offer until accepted by the 'customer'. The offer is  
good until this 'accept\_by' Date. The 'customer' must accept this Promise by this Date  
or the Promise will expire.  
Default: request.accept\_by

date\_offered - - *a Date field of model Promise*  
The Date that the supplier promised delivery to the 'customer'. Note that the Promise is  
for the 'delivery\_promises' and 'delivery\_policy' in this Promise, which may not be  
exactly what was requested by the Request.

If this Date is prior to 'request.date\_issued', then this Promise has not been offered. If  
this Date is on or after 'request.date\_issued', then this Promise has been offered as the  
answer to the 'request', and the 'request' will no longer be considered (no longer raise  
PROMISE\_NOT\_OFFERED or REQUEST\_NOT\_PLANNED Problems).  
Default: oo\_past

plan\_to\_satisfy - - *a Void field of model Promise*  
This command creates plans to satisfy this Promise. It does this by simply calling  
'plan\_to\_satisfy' on each of its 'delivery\_promises'. Thus, it is equivalent to  
'delivery\_promises.for\_each(#,plan\_to\_satisfy)'.  
Properties: command=True Export-Only Field

Alternatively, 'request.plan\_to\_satisfy' can be performed to call 'plan\_to\_satisfy' on  
each of the 'delivery\_requests'.  
Properties: command=True Export-Only Field

promise\_as\_planned, promise\_as\_planned (Logical) - - *a Void field of model Promise*  
This command sets this Promise to promise what has been planned for this Promise. It  
does this by simply calling 'promise\_as\_planned' on each of its 'delivery\_promises'.  
Thus, it is equivalent to 'delivery\_promises.for\_each(#,promise\_as\_planned)'.  
Properties: command=True Export-Only Field

plan\_as\_promised - - *a Void field of model Promise*  
This command sets the plan for this 'Promise' such that it is attempting to satisfy this  
Promise, and all of its 'delivery\_promises' and 'item\_promises'.  
Properties: command=True Export-Only Field

reject - - *a Void field of model Promise*  
This command zeros all 'item\_promises' within this Promise and sets 'offered' to 'now'.  
Properties: command=True Export-Only Field

last\_change - - *a Date field of model Promise*  
The Date at which the most recent change was made to this Promise. This is set by set-  
ting this or any other field in this model or the 'delivery\_promises'. Note that setting  
this directly will set it to the specified Date, which may not be the current Date.

Models	Promise Model
--------	---------------

This is used to support the 'changed\_promises' functions of Site, which provide net-change Exporting of Promises.  
 Default: now

problems, problems (Date\_Range) , problems (Problem\_Category) , problems (Date\_Range, Problem\_Category) -- a list(Problem)field of model Promise  
 The Problems associated with this Promise, including any problems that are associated with the underlying Delivery\_Promise and Item\_Promise models. If passed a Date\_Range, only the Problems whose 'dates' overlap are returned. If passed a Problem\_Category, only the Problems with that 'category' are returned.  
 Properties: Export-Only Field

offered -- a Date field of model Promise  
 Obsolete! This field has been renamed 'date\_offered' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.  
 Default: oo\_past

Properties: obsolete=True

owner -- a Site\_Plan field of model Promise

Properties: Export-Only Field

Models	Delivery_Promise Model
--------	------------------------

### 5.1.3.1.8.1 Delivery\_Promise Model

Delivery\_Promise -- a submodel of model Promise

Delivery\_Promise models the supplier's response to a customer's Delivery\_Request for a set of Items to be delivered together. Several Delivery\_Promises may make up the 'owner Promise.

Note that a Delivery\_Promise model exists for every Delivery\_Request model. However, the Delivery\_Promise may not be 'promised'; in fact, it may even be 'rejected'. The Delivery\_Promise models either of those responses, and the lack of a response, to the Delivery\_Request.

A Delivery\_Promise can be a response to 'actual' Delivery\_Requests from customers, or can be a response to forecasted or generated Delivery\_Requests. In either case, any commitment from the supplier is real. As 'actual' Delivery\_Requests consume from forecast Delivery\_Requests, the Delivery\_Promises corresponding to the forecast Delivery\_Requests are converted into Delivery\_Promises for the actual Delivery\_Requests. Such conversion does not require the involvement of the supplier. Thus, Delivery\_Promises made by the supplier for forecast Delivery\_Requests that have not yet been consumed by actual Delivery\_Requests represent ATP (Available-To-Promise). As such, Delivery\_Promises for forecast Delivery\_Requests are as binding as Delivery\_Promises for actual Delivery\_Requests.

The model has selectors:  
 fulfillment\_policy.

The Delivery\_Promise model has these submodels :  
 Delivery\_Available\_To\_Promise, Item\_Promise.

The Delivery\_Promise model has fields that references these models :  
 Delivery\_Request, Delivery\_Acceptance, Delivery\_Available\_To\_Promise, Item\_Promise, Problem.

These models have a field that is a Delivery\_Promise model :  
 Promise, Delivery\_Request, Delivery\_Acceptance, Delivery\_Available\_To\_Promise, REQUEST\_NOT\_PLANNED, REQUEST\_PLANNED\_LATE, REQUEST\_PLANNED\_EARLY, REQUEST\_PLANNED\_SHORT, REQUEST\_PLANNED\_EXCESS, PROMISE\_NOT\_PLANNED, PROMISE\_PLANNED\_LATE, PROMISE\_PLANNED\_EARLY, PROMISE\_PLANNED\_SHORT, PROMISE\_PLANNED\_EXCESS,

ACCEPTANCE\_NOT\_PLANNED, ACCEPTANCE\_PLANNED\_LATE, ACCEPTANCE\_PLANNED\_EARLY, ACCEPTANCE\_PLANNED\_SHORT, ACCEPTANCE\_PLANNED\_EXCESS, REQUEST\_PROMISED\_LATE, REQUEST\_PROMISED\_EARLY, REQUEST\_PROMISED\_SHORT, REQUEST\_PROMISE\_OVERPRICED, ITEM\_PROMISE\_OVERPRICED, DELIVERY\_REQUEST\_NOT\_COORDINATED, PROMISE\_NOT\_CONFIRMED, DELIVERY\_PROMISE\_NOT\_COORDINATED, DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED, SUPPLY\_PLANNED\_LATE, SUPPLY\_PLANNED\_EARLY, SUPPLY\_PLANNED\_SHORT, SUPPLY\_PLANNED\_EXCESS, SUPPLY\_PROMISED\_LATE, SUPPLY\_PROMISED\_EARLY, SUPPLY\_PROMISED\_SHORT, SUPPLY\_PROMISED\_EXCESS.

The key field for this model is `delivery_request`.  
This model may be extended with user-defined fields.

`delivery_request` - - *a Delivery\_Request field of model Delivery\_Promise*  
The `Delivery_Request` to which this `Delivery_Promise` responds.

Default: None -- this is a key field  
Properties: Export-Only Field

`delivery_acceptance` - - *a Delivery\_Acceptance field of model Delivery\_Promise*  
The `Delivery_Acceptance` for this `Delivery_Promise`.

Properties: Export-Only Field

`delivery_atp` - - *a list of Delivery\_Available\_To\_Promise submodels of model Delivery\_Promise*

The list of options for promising this `Delivery_Promise` for this Seller. For each `Item_Promise`, the list of `Item_Available_To_Promise` is computed and then sorted by delivery date. The result of this process is a list of delivery dates and what can be delivered for each `Item_Request` on that delivery date. This list can then be sorted by date at which time the first `Delivery_Available_To_Promise` will indicate the earliest date at which a delivery (it could be just a partial delivery) can be made. The last entry in the sorted list will indicate the earliest date on which the entire delivery can be made.

`promising_policy_atp` - - *a List(Delivery\_Available\_To\_Promise) field of model Delivery\_Promise*

Return the quote (subset of `delivery_atp`) according to the promising policy in effect for the Request (see `Promising_Policy` extension of `Delivery_Request`).  
Properties: Export-Only Field

`item_promises` - - *a list of Item\_Promise submodels of model Delivery\_Promise*  
Promises for the individual order-line-items.

`due` - - *a Date\_Range field of model Delivery\_Promise*  
The `Date_Range` within which the delivery is promised to be made to the 'customer'. Sometime between 'due.start' and 'due.end', the full Quantity specified in each of the 'item\_requests' should be satisfied.  
Default: forever

`rank` - - *a Number field of model Delivery\_Promise*  
The supplier-specified rank, weighing importance of this `Delivery_Promise` relative to all other `Delivery_Promise` owned by this supplier.

Note, as with all 'rank's, the higher the Number, the higher the rank, the greater the importance, the larger the weight.  
Default: 0

`fulfillment_policy` - - *an extension selector of model Delivery\_Promise*  
This policy defines whether a delivery can be shorced or delayed.

Typically this is the same as 'delivery\_request.fulfillment\_policy'. However, if the supplier is not willing to do the requested `fulfillment_policy`, the promised one may be different.

Default: delivery\_request.fulfillment\_policy

Extensions:  
`ON_TIME, FULL_QUANTITIES_OF_ALL_ITEMS, UNRESTRICTED.`

`date_confirmed` - - *a Date field of model Delivery\_Promise*  
The Date the Site confirmed this `Delivery_Promise` made by the Seller. When a Seller makes a `Promise`, it is generally based upon the availability of promises to forecast requests. Due to variance in how the forecast is modeled and reality, the Seller's `Promise` may not be feasible.

When a Site actually forms a plan to satisfy the promise and calls 'promise\_as\_planned' or 'confirm\_planned\_promises', this field is set to 'now' to indicate the `Promise` has been confirmed.  
Default: oo\_past

`list_price` - - *a Money field of model Delivery\_Promise*  
The sum of 'list\_price's for all the 'item\_promises'. This is not settable here -- it is computed from the 'item\_promises'.



Models	Delivery_Promise Model
--------	------------------------

Properties: Export-Only Field

**sum\_discount** - - *a Percentage field of model Delivery\_Promise*

The cumulative percentage discount from 'list\_price', computed from the 'item\_promises'. This is not settable here -- it is computed from the 'item\_promises'.  
Default: 0%

Properties: Export-Only Field

**sum\_price** - - *a Money field of model Delivery\_Promise*

The sum of the prices of the 'item\_promises'. This is not settable here -- it is computed from the 'item\_promises'.  
Default: list\_price

Properties: Export-Only Field

**delivery\_discount** - - *a Percentage field of model Delivery\_Promise*

The additional percentage discounted from 'sum\_price' to get the quoted 'delivery\_price'.  
Default: 0%

**delivery\_price** - - *a Money field of model Delivery\_Promise*

The price being quoted for the full delivery as specified.

Default: sum\_price

Properties: Export-Only Field

**promised\_late** - - *a Time field of model Delivery\_Promise*

If this Delivery\_Promise promises to deliver later than requested by 'item\_request', then this returns how much later; otherwise it returns "0". This is infinite if unanswered is "true". If this is non-zero and finite, then there is either a REQUEST\_PROMISED\_LATE Problem for this Delivery\_Promise. If infinite, then there is a DELIVERY\_REQUEST\_UNANSWERED Problem.

Properties: Export-Only Field

**promised\_early** - - *a Time field of model Delivery\_Promise*

If this Item\_Promise promises to deliver earlier than requested by 'item\_request', then this returns how much earlier; otherwise it returns "0". This is "0" if 'unanswered' is "true". If this is non-zero, then there is a REQUEST\_PROMISED\_EARLY Problem for this Item\_Promise.

Properties: Export-Only Field

Models	Delivery_Promise Model
--------	------------------------

**promised\_overpriced** - - *a Money field of model Delivery\_Promise*

If this Delivery\_Promise offers a 'price' that is more than the 'delivery\_request.max\_price', then this returns how much more; otherwise it returns "0". This is "0" if 'unanswered' is "true". If this is non-zero, then there is a REQUEST\_PROMISED\_OVERPRICED Problem for this Item\_Promise.

Properties: Export-Only Field

**plan\_to\_satisfy** - - *a Void field of model Delivery\_Promise*

This command sets the plan for each of this Delivery\_Promise's 'item\_promises', such that it is attempting to satisfy this promise. It is equivalent to 'item\_promises.for\_each(#plan\_to\_satisfy)'. The 'delivery\_plan.motive' will be set to match this Delivery\_Promise's 'due' Dates and its 'item\_promises' quantity's.

Alternatively, 'delivery\_request.plan\_to\_satisfy' can be performed to set the plan to attempt to satisfy the request for this promise. Or, the 'delivery\_plan.motive' can be set directly to attempt any subset of this Delivery\_Promise or its 'delivery\_request'.  
Properties: command=True Export-Only Field

**plan\_as\_promised** - - *a Void field of model Delivery\_Promise*

This command sets the plan for this Delivery\_Promise such that it is attempting to satisfy this Delivery\_Promise, and all of its 'item\_promises'.  
Properties: command=True Export-Only Field

**promise\_as\_planned** - - *a Void field of model Delivery\_Promise*

This command sets this Delivery\_Promise and its Item\_Promises according to what is currently planned (and what was requested by 'delivery\_request').

The 'due' Date\_Range will be set to include 'delivery\_request.due' and the planned Date. (The 'due.start' is set to the earlier of the 'delivery\_request.due.start' and the planned Date. The 'due.end' is set to the later of the 'delivery\_request.due.end' and the planned Date.)

Each of the 'item\_promises' quantity's are set similarly, based upon the planned Quantity's.

Properties: command=True Export-Only Field

**promise\_by\_policy** - - *a Void field of model Delivery\_Promise*

This command sets this Delivery\_Promise and its Item\_Promises according to Delivery\_Available\_To\_Promise and the promising\_policy in effect for the Delivery\_Request.

Properties: command=True Export-Only Field

Models	Delivery_Promise Model
--------	------------------------

**not\_planned** - - *a logical field of model Delivery\_Promise*  
If "true", then no plan has been formed to meet at least parts of this Delivery\_Promise. This is "true" if 'due' is finite ('unanswered' is "false") and at least one of the item\_promises' is 'unplanned'. If "true", then there is a PROMISE\_NOT\_PLANNED Problem for those unplanned item\_promises.  
Properties: Export-Only Field

**planned\_late** - - *a Time field of model Delivery\_Promise*  
Returns how late this Delivery\_Promise is currently planned to be fully delivered. It is the latest of each of the item\_promises'. Note that delivery\_promise.planned\_late' is equivalent to 'max(delivery\_promise.item\_promises.for\_each(#{planned\_late}))'. If non-zero, then there is a PROMISE\_PLANNED\_LATE Problem for at least one of the item\_promises'.

Unlike the individual item\_promises', a Delivery\_Promise can have both planned\_late' and planned\_early' non-zero.  
Properties: Export-Only Field

**planned\_early** - - *a logical field of model Delivery\_Promise*  
Returns how early this Delivery\_Promise is currently planned to be delivered, at least in part. It is the earliest of each of the item\_promises'. Note that 'delivery\_promise.planned\_early' is equivalent to 'min(delivery\_promise.item\_promises.for\_each(#{planned\_early}))'. If non-zero, then there is a PROMISE\_PLANNED\_EARLY Problem for at least one of the item\_promises'.

Unlike the individual item\_promises', a Delivery\_Promise can have both planned\_late' and planned\_early' non-zero.  
Properties: Export-Only Field

**last\_change** - - *a Date field of model Delivery\_Promise*  
The Date at which the most recent change was made to this Delivery\_Promise. This is set by setting this or any other field in this model or the item\_promises'. Note that setting this directly will set it to the specified Date, which may not be the current Date.

This is used to support the 'changed\_promises' functions of Site, which provide net-change Exporting of Promises.  
Default: now

Models	Delivery_Promise Model
--------	------------------------

problems, problems (Date\_Range) , problems (Problem\_Category) , problems (Date\_Range, Problem\_Category) - - *a List(Problem) field of model Delivery\_Promise*

The Problems associated with this Delivery\_Promise, including any problems that are associated with the underlying item\_Promise model. If passed a Date\_Range, only the Problems whose 'dates' overlap are returned. If passed a Problem\_Category, only the Problems with that 'category' are returned.  
Properties: Export-Only Field

**confirmed** - - *a Date field of model Delivery\_Promise*  
Obsolete! This field has been renamed 'date\_confirmed' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.  
Default: oo\_past  
Properties: obsolete=True

**promised\_date\_problem** - - *a Problem field of model Delivery\_Promise*  
Obsolete! Use Delivery\_Promise.problems'. The REQUEST\_PROMISED\_DATE Problem, if any, that exists between this item\_Promise and the 'item\_request' it is answering. See fields 'unanswered', 'promised\_late', and 'promised\_early' for alternate ways to test for REQUEST\_UNANSWERED, REQUEST\_PROMISED\_LATE, and REQUEST\_PROMISED\_EARLY Problems, respectively.  
Properties: obsolete=True Export-Only Field

**promised\_price\_problem** - - *a Problem field of model Delivery\_Promise*  
Obsolete! Use Delivery\_Promise.problems'. The REQUEST\_PROMISED\_OVERPRICED Problem, if any, that exists between this item\_Promise and the 'item\_request' it is answering. See the field 'overpriced' for an alternate way to test for this Problem.  
Properties: obsolete=True Export-Only Field

**plan\_problems** - - *a List(Problem) field of model Delivery\_Promise*  
Obsolete! Use Delivery\_Promise.problems'. A List of the PROMISE\_PLAN Problems of the item\_promises', if any. See fields 'unplanned', 'planned\_late', and 'planned\_early' for alternate ways to test for PROMISE\_NOT\_PLANNED, PROMISE\_PLANNED\_LATE, and PROMISE\_PLANNED\_EARLY Problems, respectively.

Note well: the lack of a Problem here only indicates that a plan to meet this Promise is being attempted. There may be feasibility Problems with the plans associated with fulfilling this Promise in addition to any Problem here.  
Properties: obsolete=True Export-Only Field

owner - - *a Promise field of model* Delivery\_Promise

Properties: Export-Only Field

5.1.3.1.8.1.1 Delivery\_Available\_To\_Promise Model

Delivery\_Available\_To\_Promise -- *a submodel of model* Delivery\_Promise

A Delivery\_Available\_To\_Promise models a forecast Promise that is suitable for the 'owner.delivery\_request', that has been allocated for use by the Seller, but has not yet been consumed by an actual (customer) Request. Such a forecast Promise is then "available" for being consumed to form a Promise for the 'owner' Delivery\_Request. This Delivery\_Available\_To\_Promise takes the form of a delivery date and a list of Item\_Available\_To\_Promises which constitute the available promises for each Item\_Request in 'owner.delivery\_request'.

Note that there may be numerous Products that match the 'owner.delivery\_request's Item\_Requests. Given that, there may be numerous different forecast Promises that are available to Promise the 'owner.delivery\_request's Item\_Request. They may vary in Quantity, delivery Date, Configuration, and price. See the Product model for more detail on how an Item\_Request is matched with a Product.

Furthermore, if the available allocation to the Seller for a particular Product is inadequate, the Seller can look at its 'organization's allocations for that Product.

Note that the "availability" recorded by these models can be fleeting. If more than one User has access to the same Seller allocations (particularly for 'organization's), then allocations can be consumed between the time that the Delivery\_Available\_To\_Promise is computed and the decision is made to consume it. Or plan revisions may be made in that same time, causing allocations to disappear. How stable ATP computations are depends upon the environment and usage. By allocating to individual Users and by insulating the 'order quoting environment' from plan changes, the ATP computations can be made very stable.

The Delivery\_Available\_To\_Promise model has fields that references these models : Delivery\_Promise.

These models have a field that is a Delivery\_Available\_To\_Promise model : Delivery\_Promise.

The key field for this model is delivery\_date

delivery\_date - - *a Date field of model* Delivery\_Available\_To\_Promise  
The delivery\_date that could be Promised with this Delivery\_Available\_To\_Promise.  
Default: None -- this is a key field  
Properties: Export-Only Field

Models	Delivery_Available_To_Promise Model
--------	-------------------------------------

item\_atp -- a List(Item\_Available\_To\_Promise) field of model

Delivery\_Available\_To\_Promise

The list of possibilities for promising 'owner:item\_promises' on 'delivery\_date' for this Seller. There will be at most one Item\_Available\_To\_Promise for each 'owner:item\_promises' to specify the ATP for that particular Item Request on 'delivery\_date'. The absence of an Item\_Available\_To\_Promise indicates that none of the Item Request can be delivered on 'delivery\_date'.

Properties: Export-Only Field

offer\_promise -- a Void field of model Delivery\_Available\_To\_Promise

Consume this Delivery\_Available\_To\_Promise in order to form a Promise for 'owner'. If this Delivery\_Available\_To\_Promise has inadequate Quantity for any of its Item\_Promises, then this will result in a new Delivery\_Request with this Delivery\_Available\_To\_Promise's Quantities. The appropriate

'owner:delivery\_request's item\_requests will be reduced in Quantity by like amount.

If the 'dates' of this satisfy the 'owner:delivery\_request', then the new Item\_Request(s) will have the same Delivery\_Request 'owner', and thus will be delivered together. However, if 'dates' does not satisfy it, then a new Delivery\_Request will be created as well.

Properties: command=True Export-Only Field

owner -- a Delivery\_Promise field of model Delivery\_Available\_To\_Promise

Properties: Export-Only Field

Models	Item_Promise Model
--------	--------------------

### 5.1.3.1.8.1.2 Item\_Promise Model

Item\_Promise -- a submodel of model Delivery\_Promise

An Item\_Promise is an agreement to supply/purchase a Quantity of a particular Item. The Date\_Range within which this Quantity should be supplied is given by the 'owner' Delivery\_Promise, which can also coordinate multiple Item\_Promises together.

The Item\_Promise model has these submodels :

Item\_Available\_To\_Promise.

The Item\_Promise model has fields that references these models :

Item\_Request, Item\_Acceptance, Item\_Available\_To\_Promise, Forecast, Configuration, Operation\_Plan, Problem, Delivery\_Promise.

These models have a field that is a Item\_Promise model :

Delivery\_Promise, Item\_Request, Item\_Acceptance, Item\_Available\_To\_Promise, REQUEST\_NOT\_PLANNED, REQUEST\_PLANNED\_LATE, REQUEST\_PLANNED\_EARLY, REQUEST\_PLANNED\_SHORT, REQUEST\_PLANNED\_EXCESS, PROMISE\_NOT\_PLANNED, PROMISE\_PLANNED\_LATE, PROMISE\_PLANNED\_EARLY, PROMISE\_PLANNED\_SHORT, PROMISE\_PLANNED\_EXCESS, ACCEPTANCE\_NOT\_PLANNED, ACCEPTANCE\_PLANNED\_LATE, ACCEPTANCE\_PLANNED\_EARLY, ACCEPTANCE\_PLANNED\_SHORT, ACCEPTANCE\_PLANNED\_EXCESS, DELIVER, REQUEST\_PROMISED\_SHORT, REQUEST\_PROMISED\_EXCESS, ITEM\_PROMISE\_OVERPRICED, SUPPLY\_PLANNED\_LATE, SUPPLY\_PLANNED\_EARLY, SUPPLY\_PLANNED\_SHORT, SUPPLY\_PLANNED\_EXCESS, SUPPLY\_PROMISED\_LATE, SUPPLY\_PROMISED\_EARLY, SUPPLY\_PROMISED\_SHORT, SUPPLY\_PROMISED\_EXCESS.

The key field for this model is item\_request

item\_request -- a Item\_Request field of model Item\_Promise

The Item\_Request that this Item\_Promise is trying to satisfy.

Default: None -- this is a key field

Properties: Export-Only Field

item\_acceptance -- a Item\_Acceptance field of model Item\_Promise

The Item\_Acceptance for this Item\_Promise.

Models	Item_Promise Model
--------	--------------------

Properties: Export-Only Field

**item\_atp** - - *a list of Item\_Available\_To\_Promise submodels of model Item\_Promise*

The list of options for promising from Item\_Available\_To\_Promise for this Seller. For each Forecast in 'matching\_forecasts', Item ATP is sought that will satisfy the 'item\_request' - that will be put in one Item\_Available\_To\_Promise model. If there is not enough Item ATP to cover the request on-time, then additional Item\_Available\_To\_Promise models will be created for the next available Dates, until the max Quantity is covered, or there is no more Item ATP.

That is repeated for each Forecast in 'matching\_forecasts' so that all options that are available to promise are listed. In that way, the best option considering price, timing, delivery\_lead\_time, and configuration can be selected.

**matching\_forecasts** - - *a List(Forecast) field of model Item\_Promise*

A List of the Forecasts that could be consumed to match 'item\_request'. Only 'active' Forecasts are considered to match (see model Forecast). The match considers four criteria: 1. requested Item must be sold as Product(\_Root).

2. the requesting customer site must match the Product(\_Root) sites, 3. the requested quantity must be greater than the Product(\_Root) min\_quantity, 4. the requested date must not be later than allowed by the Product(\_Root) min\_lead\_time, and 5. the request date must be within the effective\_dates of the Product(\_Root).

Properties: Export-Only Field

**consumed\_forecast** - - *a Forecast field of model Item\_Promise*

The Forecast, if any, that is consumed by this Item\_Promise. If nonexistent, then either this is itself a forecast Request, or this was an unexpected, unforecasted Request. The consumed\_forecast may not be a GROUP Forecast.

Default: none

**configuration** - - *a Configuration field of model Item\_Promise*

The Item being requested and the specific characteristics desired. The Item may be a STANDARD Item (no characteristics to specify), a Request-specific configuration of a OPTIONED Item, or a Request-specific CUSTOM Item (among others).

This is almost always the same as 'item\_request.configuration' (unless the requested configuration is not available or feasible). This is not settable itself, but you can set the fields of this Configuration.

Properties: command=True Export-Only Field

Models	Item_Promise Model
--------	--------------------

**item** - - *a Item field of model Item\_Promise*  
This is simply a shortcut to 'configuration.item'.

If the 'item' is [unspecified] (the default), then this is considered to be a promise for nothing.

Default: [unspecified]

**quantity** - - *a Quantity\_Range field of model Item\_Promise*

The range of Quantity of the 'configuration' that is being promised as part of the 'owner' Delivery\_Promise. This is typically 'item\_request.quantity'. However, if that Quantity was not feasible, then this Quantity may be less.

These Quantity's are converted to the 'unit' of the 'item' being requested.  
Default: 0

**list\_price** - - *a Money field of model Item\_Promise*

The standard price to be charged for the argument Item\_Request, based on its Configuration, its quantity, its due Date, its confirm\_by Date, and the delivery Location. This is used as a guideline in product quoting.

Default: 0.0

Properties: Export-Only Field

**discount** - - *a Percentage field of model Item\_Promise*

The percentage discount from 'list\_price'. Selling this sets 'price'. Selling 'price' also sets this.

Default: 0%

**price** - - *a Money field of model Item\_Promise*

The price being quoted to deliver as specified.

Default: list\_price

**promised\_short** - - *a Quantity field of model Item\_Promise*

If this Item\_Promise promises less than requested by 'item\_request', then this returns how much less; otherwise it returns "0". This is "0" if 'quantity' is "0" or 'owner.due' is infinite. If this is non-zero, then there is a REQUEST\_PROMISED\_SHORT Problem for this Item\_Promise.

Properties: Export-Only Field

**promised\_excess** - - *a Quantity field of model Item\_Promise*

If this Item\_Promise promises more than requested by Item\_request, then this returns how much more; otherwise it returns "0". This is "0" if 'quantity' is "0" or 'owner.due' is infinite. If this is non-zero, then there is a REQUEST\_PROMISED\_EXCESS Problem for this Item\_Promise.

Properties: Export-Only Field

**promised\_overpriced** - - *a Money field of model Item\_Promise*

If this Item\_Promise offers a 'price' that is more than the Item\_request.max\_price, then this returns how much more; otherwise it returns "0". This is "0" if 'quantity' is "0" or 'owner.due' is infinite. If this is non-zero, then there is a REQUEST\_PROMISED\_OVERPRICED Problem for this Item\_Promise.

Properties: Export-Only Field

**delivery\_plan** - - *a Operation\_Plan field of model Item\_Promise*

The Operation that has been planned to deliver this Item\_Promise. It is the same as Item\_request.delivery\_plan.

Properties: Export-Only Field

**plan\_to\_satisfy** - - *a Void field of model Item\_Promise*

This command sets the plan for this Item\_Promise such that it is attempting to satisfy this promise. The 'delivery\_plan.motive.planned\_for\_promise' will be set "true" which will cause it to try to satisfy this Item\_Promise's 'quantity' and its 'owner' Delivery\_Promise's 'due' Dates.

Alternatively, Item\_request.plan\_to\_satisfy can be performed to set the plan to attempt to satisfy the request rather than the promise. Or, the planned\_for\_promise field of the 'delivery\_plan.motive' can be set directly to select whether to plan for this promise or the Item\_request.

Properties: command=True Export-Only Field

**promise\_as\_planned** - - *a Void field of model Item\_Promise*

This command sets this Item\_Promise's 'quantity' according to the Quantity currently planned and what was requested.

The 'quantity' is set to the Quantity\_Range that includes all of Item\_request.quantity and the planned Quantity. (The 'quantity.min' is set to the lesser of the Item\_request.quantity.min and the planned Quantity. The 'quantity.max' is set to the greater of the Item\_request.quantity.max and the planned Quantity.)

Properties: command=True Export-Only Field

**plan\_as\_promised** - - *a Void field of model Item\_Promise*

This command creates or re-sizes the receiving\_plan for this Item\_RAP such that it satisfies this promise.

Properties: command=True Export-Only Field

**receiving\_plan** - - *a Operation\_Plan field of model Item\_Promise*

The Operation that has been planned to receive this Item\_Promise.

Properties: Export-Only Field

**not\_planned** - - *a Logical field of model Item\_Promise*

If "true", then no Operation has been planned to satisfy this Item\_Promise. If "true", then there is a PROMISE\_NOT\_PLANNED Problem for this Item\_Promise.

Properties: Export-Only Field

**planned\_late** - - *a Time field of model Item\_Promise*

If the plan for this Item\_Promise is planned for later than promised, then this returns how much later; otherwise it returns "0". This is "0" if 'quantity' is "0" or 'owner.due' is infinite. If this is non-zero, then there is a PROMISE\_PLANNED\_LATE Problem for this Item\_Promise.

Properties: Export-Only Field

**planned\_early** - - *a Time field of model Item\_Promise*

If the plan for this Item\_Promise is planned for earlier than promised, then this returns how much earlier; otherwise it returns "0". This is "0" if 'quantity' is "0" or 'owner.due' is infinite. If this is non-zero, then there is a PROMISE\_PLANNED\_EARLY Problem for this Item\_Promise.

Properties: Export-Only Field

**planned\_short** - - *a Quantity field of model Item\_Promise*

If the plan for this Item\_Promise is planned to deliver less than promised, then this returns how much less; otherwise it returns "0". This is "0" if 'quantity' is "0" or 'owner.due' is infinite. If this is non-zero, then there is a PROMISE\_PLANNED\_SHORT Problem for this Item\_Promise.

Properties: Export-Only Field

**planned\_excess** - - *a Quantity field of model Item\_Promise*

If the plan for this Item\_Promise is planned to deliver more than promised, then this returns how much more; otherwise it returns "0". This is "0" if 'quantity' is "0" or 'owner.due' is infinite. If this is non-zero, then there is a PROMISE\_PLANNED\_EXCESS Problem for this Item\_Promise.

Properties: Export-Only Field

Models	Item_Promise Model
--------	--------------------

last\_change - - a Date field of model Item\_Promise  
 The Date at which the most recent change was made to this Item\_Promise. This is set by setting this or any other field in this model. Note that setting this directly will set it to the specified Date, which may not be the current Date.

This is used to support the changed\_promises' functions of Site, which provide net-change exporting of Promises.

This is the same as calling owner.last\_change.  
 Default: now

problems, problems(Date\_Range) , problems(Problem\_Category) , problems(Date\_Range, Problem\_Category) - - a List(Problem) field of model Item\_Promise

The Problems associated with this Item\_Promise If passed a Date\_Range, only the Problems whose dates overlap are returned. If passed a Problem\_Category, only the Problems with that category are returned.

Properties: Export-Only Field

promised\_quantity\_problem - - a Problem field of model Item\_Promise

Obsolete! Use Item\_Promise.problems: The REQUEST\_PROMISED\_QUANTITY Problem, if any, that exists between this Item\_Promise and the item\_request' it is answering. See fields promised\_short' and promised\_excess' for more convenient ways to test for REQUEST\_PROMISED\_SHORT, and REQUEST\_PROMISED\_EXCESS Problems, respectively. This will be nonexistent if there is no such Problem or if owner.unanswered' is "true".

Properties: obsolete=True Export-Only Field

promised\_price\_problem - - a Problem field of model Item\_Promise

Obsolete! Use Item\_Promise.problems: The REQUEST\_PROMISED\_OVERPRICED Problem, if any, that exists between this Item\_Promise and the item\_request' it is answering. See the field 'overpriced' for an alternate way to test for this Problem. This will be nonexistent if there is no such Problem or if owner.unanswered' is "true".

Properties: obsolete=True Export-Only Field

Models	Item_Promise Model
--------	--------------------

planned\_date\_problem - - a Problem field of model Item\_Promise  
 Obsolete! Use Item\_Promise.problems: The PROMISE\_PLANNED\_DATE Problem, if any, that exists between this Item\_Promise and the Operation\_Plan currently planned to satisfy it (delivery\_plan). See fields 'not\_planned', planned\_late', and planned\_early' for alternate ways to test for PROMISE\_NOT\_PLANNED, PROMISE\_PLANNED\_LATE, and PROMISE\_PLANNED\_EARLY Problems, respectively.

Note well: the lack of a Problem here only indicates that a plan to meet this Promise is being attempted. There may be feasibility Problems with the plans associated with fulfilling this Promise in addition to any Problem here.

Properties: obsolete=True Export-Only Field

planned\_quantity\_problem - - a Problem field of model Item\_Promise

Obsolete! Use Item\_Promise.problems: The PROMISE\_PLANNED\_QUANTITY Problem, if any, that exists between this Item\_Promise and the Operation\_Plan currently planned to satisfy it (delivery\_plan). See fields 'not\_planned', planned\_short', and 'planned\_excess' for more convenient ways to test for PROMISE\_NOT\_PLANNED, PROMISE\_PLANNED\_SHORT, and PROMISE\_PLANNED\_EXCESS Problems, respectively.

Note well: the lack of a Problem here only indicates that a plan to meet this Promise is being attempted. There may be feasibility Problems with the plans associated with fulfilling this Promise in addition to any Problem here.

Properties: obsolete=True Export-Only Field

owner - - a Delivery\_Promise field of model Item\_Promise

Properties: Export-Only Field

Models	Item_Available_To_Promise Model
--------	---------------------------------

### 5.1.3.1.8.1.2.1 Item\_Available\_To\_Promise Model

Item\_Available\_To\_Promise -- *a submodel of model Item\_Promise*

An Item\_Available\_To\_Promise models the available quantity of an item on a specific date (the Delivery\_Available\_To\_Promise date) to satisfy an Item\_Request for the requested item. Therefore, it models a forecast Promise that is suitable for the owner.item\_request; that has been allocated for use by the Seller, but has not yet been consumed by an actual (customer) Request. Such a forecast Promise is then "available" for being consumed to form a Promise for the owner.Item\_Request. This Item\_Available\_To\_Promise takes the form of a item date and a list of Item\_Available\_To\_Promises which constitute the available promises for each Item\_Request in 'owner.item\_request'.

Note that there may be numerous Products that match the 'owner.item\_request's Item\_Requests. Given that, there may be numerous different forecast Promises that are available to Promise the 'owner.item\_request's Item\_Request. They may vary in Quantity, item Date, Configuration, and price. See the Product model for more detail on how an Item\_Request is matched with a Product.

Furthermore, if the available allocation to the Seller for a particular Product is inadequate, the Seller can look at its organization's allocations for that Product.

Note that the "availability" recorded by these models can be fleeting. If more than one User has access to the same Seller allocations (particularly for 'organization's), then allocations can be consumed between the time that the Item\_Available\_To\_Promise is computed and the decision is made to consume it. Or plan revisions may be made in that same time, causing allocations to disappear. How stable ATP computations are depends upon the environment and usage. By allocating to individual Users and by insulating the 'order quoting environment' from plan changes, the ATP computations can be made very stable.

The Item\_Available\_To\_Promise model has these submodels :  
Product\_Available\_To\_Promise.

The Item\_Available\_To\_Promise model has fields that references these models :  
Product\_Available\_To\_Promise, Item\_Promise.

These models have a field that is a Item\_Available\_To\_Promise model :  
Item\_Promise, Product\_Available\_To\_Promise.

The key field for this model is dates

Models	Item_Available_To_Promise Model
--------	---------------------------------

dates -- *a Date\_Range field of model Item\_Available\_To\_Promise*  
The due Date\_Range that could be Promised with this Item\_Available\_To\_Promise.  
Default: None -- this is a key field  
Properties: Export-Only Field

available -- *a Quantity field of model Item\_Available\_To\_Promise*  
The amount available to promise for 'owner.item\_request'. This quantity is computed according to the following rule: min(owner.item\_request.quantity,min,product\_atp.for\_each(#.quantity),sum)  
Properties: Export-Only Field

product\_atp -- *a list of Product\_Available\_To\_Promise submodels of model Item\_Available\_To\_Promise*  
The list of possibilities for promising 'owner' on 'dates' for this Seller.

offer\_promise -- *a Void field of model Item\_Available\_To\_Promise*  
Consume this Item\_Available\_To\_Promise in order to form a Promise for 'owner'. If this Item\_Available\_To\_Promise has inadequate Quantity for its 'owner', then this will result in a new Item\_Request with this Item\_Available\_To\_Promise's 'available' quantity. The appropriate 'owner.item\_request's item\_request will be reduced in Quantity by like amount.

If the 'dates' of this satisfy the 'owner.item\_request', then the new Item\_Request(s) will have the same Item\_Request 'owner', and thus will be delivered together. However, if 'dates' does not satisfy it, then a new Item\_Request will be created as well.  
Properties: command=True Export-Only Field

owner -- *a Item\_Promise field of model Item\_Available\_To\_Promise*  
Properties: Export-Only Field



5.1.3.1.8.1.2.1.1 Product\_Available\_To\_Promise Model

Product\_Available\_To\_Promise -- a submodel of model Item\_Available\_To\_Promise

A Product\_Available\_To\_Promise models the quantity available for each equivalent product, since more than one product may be able to satisfy the request for the item. Therefore, a forecast Promise that is suitable for the 'owner' Item\_Request, that has been allocated for use by the Seller, but has not yet been consumed by an actual (customer) Request. Such a forecast Promise is then "available" for being consumed to form a Promise for the 'owner' Item\_Request.

Note that there may be numerous Products that match the 'owner' Item\_Request. Given that, there may be numerous different forecast Promises that are available to Promise the 'owner' Item\_Request. They may vary in Quantity, delivery Date, Configuration, and price. See the Product model for more detail on how an Item\_Request is matched with a Product.

Furthermore, if the available allocation to the Seller for a particular Product is inadequate, the Seller can look at its 'organization's allocations for that Product.

Note that the "availability" recorded by these models can be fleeting. If more than one User has access to the same Seller allocations (particularly for 'organization's), then allocations can be consumed between the time that the Product\_Available\_To\_Promise is computed and the decision is made to consume it. Or plan revisions may be made in that same time, causing allocations to disappear. How stable ATP computations are depends upon the environment and usage. By allocating to individual Users and by insulating the order quoting environment from plan changes, the ATP computations can be made very stable.

The Product\_Available\_To\_Promise model has fields that references these models : Product, Forecast, Forecast\_Entry, Item\_Available\_To\_Promise.

These models have a field that is a Product\_Available\_To\_Promise model : Item\_Available\_To\_Promise.

The key field for this model is product

product -- a Product field of model Product\_Available\_To\_Promise  
The Product for which there is this Product\_Available\_To\_Promise.

This is not really settable (doc to be fixed).  
Default: None -- this is a key field

Properties: Export-Only Field

forecast -- a Forecast field of model Product\_Available\_To\_Promise  
The Forecast for 'product' from which this Product\_Available\_To\_Promise was computed.  
Properties: Export-Only Field

forecast\_entry -- a Forecast\_Entry field of model Product\_Available\_To\_Promise  
The latest Forecast\_Entry for 'product' from which this Product\_Available\_To\_Promise was computed.  
Properties: Export-Only Field

dates -- a Date\_Range field of model Product\_Available\_To\_Promise  
The due Date\_Range that could be Promised with this Product\_Available\_To\_Promise.  
Properties: Export-Only Field

price -- a Money field of model Product\_Available\_To\_Promise  
The price that could be quoted for this Product\_Available\_To\_Promise.  
Properties: Export-Only Field

quantity -- a Quantity field of model Product\_Available\_To\_Promise  
The Quantity that is Product\_Available\_To\_Promise to the 'owner' Item\_Request from the 'forecast\_entry'. This Quantity is converted to the 'unit' of the 'item' of the 'owner' Item\_Request.  
Properties: Export-Only Field

offered\_quantity -- a Quantity field of model Product\_Available\_To\_Promise  
The Quantity that will be promised to the 'owner' Item\_Request by 'offer\_promise'. This Quantity is converted to the 'unit' of the 'item' of the 'owner' Item\_Request.  
Default: value->available()

offer\_promise, offer\_promise (Quantity) -- a Void field of model Product\_Available\_To\_Promise  
Consume this Product\_Available\_To\_Promise in order to form a Promise for 'owner'. If this Item\_Available\_To\_Promise has inadequate Quantity, then this will result in a new Item\_Request with this Item\_Available\_To\_Promise's Quantity. The 'owner' Item\_Request will be reduced in Quantity by like amount.

If the 'dates' of this satisfy the 'owner's Delivery\_Request, then the new Item\_Request will have the same Delivery\_Request owner', and thus will be delivered together. However, if 'dates' does not satisfy it, then a new Delivery\_Request will be created as well.

The 'owner.item\_request.quantity.max' will be reduced by the Quantity consumed to satisfy the 'owner'. Of course, no more than 'owner.item\_request.quantity.max' will be consumed, so 'owner.item\_request.quantity.max' will never be less than 0.

Properties: command=True Export-Only Field

owner -- a Item\_Available\_To\_Promise field of model  
Product\_Available\_To\_Promise

Properties: Export-Only Field  
5.1.3.1.9 Acceptance Model

Acceptance -- a submodel of model Site\_Plan

An Acceptance models the 'owner' requester's response to a 'supplier's Promise. Note that a Acceptance model exists for every Request/Promise model. However, the Acceptance may not be 'accepted' yet.

The Acceptance model has these submodels :  
Delivery\_Acceptance.

The Acceptance model has fields that references these models :  
Promise, Delivery\_Acceptance, Site\_Plan.

These models have a field that is a Acceptance model :

LINK, Promise, SUPPLIER, Delivery\_Acceptance,  
REQUEST\_NOT\_PLANNED, REQUEST\_PLANNED\_LATE,  
REQUEST\_PLANNED\_EARLY, REQUEST\_PLANNED\_SHORT,  
REQUEST\_PLANNED\_EXCESS, PROMISE\_NOT\_PLANNED,  
PROMISE\_PLANNED\_LATE, PROMISE\_PLANNED\_EARLY,  
PROMISE\_PLANNED\_SHORT, PROMISE\_PLANNED\_EXCESS,  
ACCEPTANCE\_NOT\_PLANNED, ACCEPTANCE\_PLANNED\_LATE,  
ACCEPTANCE\_PLANNED\_EARLY, ACCEPTANCE\_PLANNED\_SHORT,  
ACCEPTANCE\_PLANNED\_EXCESS, REQUEST\_PROMISED\_LATE,  
REQUEST\_PROMISED\_EARLY, REQUEST\_PROMISED\_SHORT,  
REQUEST\_PROMISED\_EXCESS, ITEM\_PROMISE\_OVERPRICED,  
DELIVERY\_PROMISE\_OVERPRICED, PROMISE\_NOT\_OFFERED,  
PROMISE\_NOT\_CONFIRMED, PROMISE\_NOT\_ACCEPTED,

ACCEPTANCE\_INCONSISTENT, REQUEST\_QUEUED,  
DELIVERY\_REQUEST\_NOT\_COORDINATED,  
DELIVERY\_PROMISE\_NOT\_COORDINATED,  
DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED,  
SUPPLY\_PLANNED\_LATE, SUPPLY\_PLANNED\_EARLY,  
SUPPLY\_PLANNED\_SHORT, SUPPLY\_PLANNED\_EXCESS,  
SUPPLY\_PROMISED\_LATE, SUPPLY\_PROMISED\_EARLY,  
SUPPLY\_PROMISED\_SHORT, SUPPLY\_PROMISED\_EXCESS.

The key field for this model is promise  
This model may be extended with user-defined fields.

promise -- a Promise field of model Acceptance  
The Promise to which this Acceptance responds.

Default: None -- this is a key field

Properties: Export-Only Field

delivery\_acceptances -- a list of Delivery\_Acceptance submodels of model Acceptance  
The individual acceptances for delivery\_promises.

accepted -- a Date field of model Acceptance  
The Date that the requester accepted the promise  
Default: oo\_past

accept\_by -- a Date field of model Acceptance  
The Date by which the 'customer' intends to accept or reject any Promise that is made.  
This is the Date on which both parties would become committed. And it is the start of the delivery\_lead\_time (which can affect pricing).

Note that this is just the requested 'accept\_by' Date -- the actual deadline for accepting a Promise is in the Promise 'accept\_by'. The supplier considers this requested Date when making the Promise, but may choose a different Date based upon the delivery lead times of its Products. Such differences are part of the negotiation of Request and Promise.

Default: promise.request.promise\_by + "1 day"

plan\_to\_satisfy -- a Void field of model Acceptance

This command creates plans to satisfy this Acceptance. It does this by simply calling 'plan\_to\_satisfy' on each of its 'delivery\_acceptances'. Thus, it is equivalent to 'delivery\_acceptances.for\_each(#.plan\_to\_satisfy)'.  
plan\_to\_satisfy

Alternatively, 'request.plan\_to\_satisfy' can be performed to call 'plan\_to\_satisfy' on each of the 'delivery\_requests'.

Properties: command=True Export-Only Field

plan\_as\_accepted -- a Void field of model Acceptance

This command sets the plan for each of this Acceptance's 'item\_acceptances'.

Properties: command=True Export-Only Field

accept\_as\_promised, accept\_as\_promised (Logical) -- a Void field of model Acceptance

This command sets this Acceptance to accept what has been promised. It does this by simply calling 'accept\_as\_promised' on each of its 'delivery\_acceptance'. Thus, it is equivalent to 'delivery\_acceptances.for\_each(#accept\_as\_promised)'.

Properties: command=True Export-Only Field

last\_change -- a Date field of model Acceptance

The Date at which the most recent change was made to this Acceptance. This is set by setting this or any other field in this model or the 'delivery\_acceptances'. Note that setting this directly will set it to the specified Date, which may not be the current Date.

This is used to support the 'changed\_acceptances' functions of Site, which provide net-change Exporting of Acceptances.

Default: now

problems, problems (Date\_Range) , problems (Problem\_Category) , problems (Date\_Range, Problem\_Category) -- a List(Problem) field of model Acceptance  
The Problems associated with this Acceptance, including any problems that are associated with the underlying 'Delivery\_Acceptance' and 'Item\_Acceptance' models. If passed a 'Date\_Range', only the Problems whose 'dates' overlap are returned. If passed a 'Problem\_Category', only the Problems with that 'category' are returned.

Properties: Export-Only Field

owner -- a Site\_Plan field of model Acceptance

Properties: Export-Only Field

### 5.1.3.1.9.1 Delivery\_Acceptance Model

Delivery\_Acceptance -- a submodel of model Acceptance

Delivery\_Acceptance models the supplier's response to a 'customer's Delivery\_Request' for a set of Items to be delivered together. Several 'Delivery\_Acceptances' may make up the 'owner' 'Acceptance'.

The model has selectors:  
fulfillment\_policy.

The 'Delivery\_Acceptance' model has these submodels:  
Item\_Acceptance.

The 'Delivery\_Acceptance' model has fields that references these models:  
Delivery\_Promise, Item\_Acceptance, Problem.

These models have a field that is a 'Delivery\_Acceptance' model:  
Acceptance, Delivery\_Promise, Item\_Acceptance,  
REQUEST\_NOT\_PLANNED, REQUEST\_PLANNED\_LATE,  
REQUEST\_PLANNED\_EARLY, REQUEST\_PLANNED\_SHORT,  
REQUEST\_PLANNED\_EXCESS, PROMISE\_NOT\_PLANNED,  
PROMISE\_PLANNED\_LATE, PROMISE\_PLANNED\_EARLY,  
PROMISE\_PLANNED\_SHORT, PROMISE\_PLANNED\_EXCESS,  
ACCEPTANCE\_NOT\_PLANNED, ACCEPTANCE\_PLANNED\_LATE,  
ACCEPTANCE\_PLANNED\_EARLY, ACCEPTANCE\_PLANNED\_SHORT,  
ACCEPTANCE\_PLANNED\_EXCESS, REQUEST\_PROMISED\_LATE,  
REQUEST\_PROMISED\_EARLY, REQUEST\_PROMISED\_SHORT,  
REQUEST\_PROMISED\_EXCESS, ITEM\_PROMISE\_OVERPRICED,  
DELIVERY\_PROMISE\_OVERPRICED, PROMISE\_NOT\_CONFIRMED,  
DELIVERY\_REQUEST\_NOT\_COORDINATED,  
DELIVERY\_PROMISE\_NOT\_COORDINATED,  
DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED,  
SUPPLY\_PLANNED\_LATE, SUPPLY\_PLANNED\_EARLY,  
SUPPLY\_PLANNED\_SHORT, SUPPLY\_PLANNED\_EXCESS,  
SUPPLY\_PROMISED\_LATE, SUPPLY\_PROMISED\_EARLY,  
SUPPLY\_PROMISED\_SHORT, SUPPLY\_PROMISED\_EXCESS.

The key field for this model is 'delivery\_promise'.  
This model may be extended with user-defined fields.

*delivery\_promise* - - *a Delivery\_Promise field of model Delivery\_Acceptance*  
The *Delivery\_Promise* to which this *Delivery\_Acceptance* responds.  
Default: None -- this is a key field  
Properties: Export-Only Field

*item\_acceptances* - - *a list of Item\_Acceptance submodels of model Delivery\_Acceptance*  
Acceptances for the individual order-line-items.

*due* - - *a Date\_Range field of model Delivery\_Acceptance*  
The *Date\_Range* within which the promise is accepted by the 'customer'.  
Default: *delivery\_promise.due*

*fulfillment\_policy* - - *an extension selector of model Delivery\_Acceptance*  
This policy defines whether a delivery can be shotted or delayed.

Typically this is the same as '*delivery\_promise.fulfillment\_policy*'. However, if the customer is not willing to do the promised '*fulfillment\_policy*', the acceptance one may be different.

Default: *delivery\_promise.fulfillment\_policy*  
Extensions:  
**ON\_TIME, FULL\_QUANTITIES\_OF\_ALL\_ITEMS, UNRESTRICTED.**

*plan\_to\_satisfy* - - *a Void field of model Delivery\_Acceptance*  
This command sets the plan for each of this *Delivery\_Acceptance*'s '*item\_acceptances*'.  
Properties: command=True Export-Only Field

*plan\_as\_accepted* - - *a Void field of model Delivery\_Acceptance*  
This command sets the plan for each of this *Delivery\_Acceptance*'s '*item\_acceptances*'.  
Properties: command=True Export-Only Field

*accept\_as\_promised* - - *a Void field of model Delivery\_Acceptance*  
This command sets this *Delivery\_Acceptance* and its '*item\_acceptances*' according to what is currently promised  
Properties: command=True Export-Only Field

*not\_planned* - - *a Logical field of model Delivery\_Acceptance*  
If "true", then no plan has been formed to meet at least parts of this *Delivery\_Acceptance*. This is "true" if '*due*' is finite ('unanswered' is "false") and at least one of the '*item\_acceptances*' is 'unplanned'. If "true", then there is a **ACCEPTANCE\_NOT\_PLANNED** Problem for those unplanned '*Item\_Promises*'.  
Properties: Export-Only Field

*planned\_late* - - *a Time field of model Delivery\_Acceptance*  
Returns how late this *Delivery\_Acceptance* is currently planned to be fully delivered. It is the latest of each of the '*item\_acceptances*'. Note that '*delivery\_acceptance.planned\_late*' is equivalent to '*max(delivery\_acceptance.item\_acceptances.for\_each(#.planned\_late))*'. If non-zero, then there is a **ACCEPTANCE\_PLANNED\_LATE** Problem for at least one of the '*item\_acceptances*'.

Unlike the individual '*item\_acceptances*', a *Delivery\_Acceptance* can have both '*planned\_late*' and '*planned\_early*' non-zero.  
Properties: Export-Only Field

*planned\_early* - - *a Logical field of model Delivery\_Acceptance*  
Returns how early this *Delivery\_Acceptance* is currently planned to be delivered, at least in part. It is the earliest of each of the '*item\_acceptances*'. Note that '*delivery\_acceptance.planned\_early*' is equivalent to '*min(delivery\_acceptance.item\_acceptances.for\_each(#.planned\_early))*'. If non-zero, then there is a **ACCEPTANCE\_PLANNED\_EARLY** Problem for at least one of the '*item\_acceptances*'.

Unlike the individual '*item\_acceptances*', a *Delivery\_Acceptance* can have both '*planned\_late*' and '*planned\_early*' non-zero.  
Properties: Export-Only Field

*last\_change* - - *a Date field of model Delivery\_Acceptance*  
The Date at which the most recent change was made to this *Delivery\_Acceptance*. This is set by setting this or any other field in this model or the '*item\_acceptances*'. Note that setting this directly will set it to the specified Date, which may not be the current Date.

This is used to support the '*changed\_acceptances*' functions of *Site*, which provide net-change Exporting of Acceptances.  
Default: now

problems, problems (Date\_Range) , problems (Problem\_Category) , problems (Date\_Range, Problem\_Category) -- a List(Problem)field of model Delivery\_Acceptance

The Problems associated with this Delivery\_Acceptance, including any problems that are associated with the underlying Item\_Acceptance model. If passed a Date\_Range, only the Problems whose 'dates' overlap are returned. If passed a Problem\_Category, only the Problems with that 'category' are returned.

Properties: Export-Only Field

plan\_problems -- a List(Problem)field of model Delivery\_Acceptance  
Obsolete! Use Delivery\_Acceptance 'problems'. A List of the ACCEPTANCE\_PLAN Problems of the Item\_Acceptances, if any. See fields 'unplanned', 'planned\_late', and 'planned\_early' for alternate ways to test for ACCEPTANCE\_NOT\_PLANNED, ACCEPTANCE\_PLANNED\_LATE, and ACCEPTANCE\_PLANNED\_EARLY Problems, respectively.

Note well: the lack of a Problem here only indicates that a plan to meet this Acceptance is being attempted. There may be feasibility Problems with the plans associated with fulfilling this Acceptance in addition to any Problem here.

Properties: obsolete=True Export-Only Field

delivery\_not\_coordinated\_problem -- a Problemfield of model Delivery\_Acceptance  
Obsolete! Use Delivery\_Acceptance 'problems'. The DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED Problem, if any, that exists

Properties: obsolete=True Export-Only Field

owner -- a Acceptancefield of model Delivery\_Acceptance

Properties: Export-Only Field

### 5.1.3.1.9.1.1 Item\_Acceptance Model

Item\_Acceptance -- a submodel of model Delivery\_Acceptance

An Item\_Acceptance is an agreement to supply/consume a Quantity of a particular Item. The Date\_Range within which this Quantity should be supplied is given by the 'owner' Delivery\_Acceptance, which can also coordinate multiple Item\_Acceptances together.

The Item\_Acceptance model has fields that references these models :  
Item\_Promise, Configuration, Operation\_Plan, Problem, Delivery\_Acceptance.

These models have a field that is a Item\_Acceptance model :  
Delivery\_Acceptance, Item\_Promise, REQUEST\_NOT\_PLANNED, REQUEST\_PLANNED\_LATE, REQUEST\_PLANNED\_EARLY, REQUEST\_PLANNED\_SHORT, REQUEST\_PLANNED\_EXCESS, PROMISE\_NOT\_PLANNED, PROMISE\_PLANNED\_LATE, PROMISE\_PLANNED\_EARLY, PROMISE\_PLANNED\_SHORT, PROMISE\_PLANNED\_EXCESS, ACCEPTANCE\_NOT\_PLANNED, ACCEPTANCE\_PLANNED\_LATE, ACCEPTANCE\_PLANNED\_EARLY, ACCEPTANCE\_PLANNED\_SHORT, ACCEPTANCE\_PLANNED\_EXCESS, REQUEST\_PROMISED\_SHORT, REQUEST\_PROMISED\_EXCESS, ITEM\_PROMISE\_OVERRICED, SUPPLY\_PLANNED\_LATE, SUPPLY\_PLANNED\_EARLY, SUPPLY\_PLANNED\_SHORT, SUPPLY\_PLANNED\_EXCESS, SUPPLY\_PROMISED\_LATE, SUPPLY\_PROMISED\_EARLY, SUPPLY\_PROMISED\_SHORT, SUPPLY\_PROMISED\_EXCESS.

The key field for this model is item\_promise

item\_promise -- a Item\_Promisefield of model Item\_Acceptance  
The Item\_Request that this Item\_Promise is trying to satisfy.

Default: None -- this is a key field  
Properties: Export-Only Field

configuration -- a Configurationfield of model Item\_Acceptance  
The Item being requested and the specific characteristics desired. The Item may be a STANDARD Item (no characteristics to specify), a Request-specific configuration of a OPTIONED Item, or a Request-specific CUSTOM Item (among others).

This is almost always the same as 'Item\_request.configuration' (unless the requested configuration is not available or feasible). This is not scilable itself, but you can set the fields of this Configuration.

Properties:    command=True    Export-Only Field

Item - - *a Item field of model Item\_Acceptance*

This is simply a shortcut to 'configuration.item'.

If the 'item' is [unspecified] (the default), then this is considered to be a promise for nothing.

Default:    [unspecified]

quantity - - *a Quantity\_Range field of model Item\_Acceptance*

The range of Quantity of the 'configuration' that is being accepted as part of the 'owner' Delivery\_Acceptance. This is typically 'Item\_request.quantity'. However, if that Quantity was not feasible, then this Quantity may be less.

These Quantity's are converted to the 'unit' of the 'item' being requested.  
Default:    'item.promise.quantity'

delivery\_plan - - *a Operation\_Plan field of model Item\_Acceptance*

The Operation that has been planned to deliver this Item\_Acceptance. It is the same as 'Item\_request.delivery\_plan'.

Properties:    Export-Only Field

plan\_to\_satisfy - - *a Void field of model Item\_Acceptance*

This command sets the plan for this Item\_Acceptance such that it is attempting to satisfy this acceptance. The 'delivery\_plan.motive.planned\_for\_acceptance' will be set "true" which will cause it to try to satisfy this Item\_Acceptance's 'quantity' and its 'owner' Delivery\_Acceptance's 'due' Dates.

Properties:    command=True    Export-Only Field

accept\_as\_promised - - *a Void field of model Item\_Acceptance*

This command sets this Item\_Acceptance's 'quantity' according to the Quantity currently planned and what was requested.

The 'quantity' is set to the Quantity\_Range that includes all of 'Item\_request.quantity' and the planned Quantity. (The 'quantity.min' is set to the lesser of the 'Item\_request.quantity.min' and the planned Quantity. The 'quantity.max' is set to the greater of the 'Item\_request.quantity.max' and the planned Quantity.)  
Properties:    command=True    Export-Only Field

plan\_as\_accepted - - *a Void field of model Item\_Acceptance*  
This command creates or re-sizes the receiving\_plan for this Item\_RAP such that it satisfies this acceptance.

Properties:    command=True    Export-Only Field

not\_planned - - *a Logical field of model Item\_Acceptance*

If "true", then no Operation has been planned to satisfy this Item\_Acceptance. If "true", then there is a ACCEPTANCE\_NOT\_PLANNED Problem for this Item\_Acceptance.

Properties:    Export-Only Field

planned\_late - - *a Time field of model Item\_Acceptance*

If the plan for this Item\_Acceptance is planned for later than accepted, then this returns how much later; otherwise it returns "0". This is "0" if 'quantity' is "0" or 'owner.due' is infinite. If this is non-zero, then there is a ACCEPTANCE\_PLANNED\_LATE Problem for this Item\_Acceptance.

Properties:    Export-Only Field

planned\_early - - *a Time field of model Item\_Acceptance*

If the plan for this Item\_Acceptance is planned for earlier than accepted, then this returns how much earlier; otherwise it returns "0". This is "0" if 'quantity' is "0" or 'owner.due' is infinite. If this is non-zero, then there is a ACCEPTANCE\_PLANNED\_EARLY Problem for this Item\_Acceptance.

Properties:    Export-Only Field

planned\_short - - *a Quantity field of model Item\_Acceptance*

If the plan for this Item\_Acceptance is planned to deliver less than accepted, then this returns how much less; otherwise it returns "0". This is "0" if 'quantity' is "0" or 'owner.due' is infinite. If this is non-zero, then there is a ACCEPTANCE\_PLANNED\_SHORT Problem for this Item\_Acceptance.

Properties:    Export-Only Field

planned\_excess - - *a Quantity field of model Item\_Acceptance*

If the plan for this Item\_Acceptance is planned to deliver more than accepted, then this returns how much more; otherwise it returns "0". This is "0" if 'quantity' is "0" or 'owner.due' is infinite. If this is non-zero, then there is a ACCEPTANCE\_PLANNED\_EXCESS Problem for this Item\_Acceptance.

Properties:    Export-Only Field

Models	Item_Acceptance Model
--------	-----------------------

last\_change - - a Date field of model Item\_Acceptance  
 The Date at which the most recent change was made to this Item\_Acceptance. This is set by setting this or any other field in this model. Note that setting this directly will set it to the specified Date, which may not be the current Date.

This is used to support the 'changed\_acceptances' functions of Site, which provide net-change exporting of Acceptances.

This is the same as calling owner.r\_last\_change.  
 Default: now

problems, problems (Date\_Range) , problems (Problem\_Category) , problems (Date\_Range, Problem\_Category) - - a List[Problem] field of model Item\_Acceptance

The Problems associated with this Item\_Acceptance If passed a Date\_Range, only the Problems whose 'dates' overlap are returned. If passed a Problem\_Category, only the Problems with that 'category' are returned.

Properties: Export-Only Field

planned\_date\_problem - - a Problem field of model Item\_Acceptance  
 Obsolete! Use Item\_Acceptance 'problems'. The ACCEPTANCE\_PLANNED\_DATE Problem, if any, that exists between this Item\_Acceptance and the Operation\_Plan currently planned to satisfy it ('delivery\_plan'). See fields 'not\_planned', 'planned\_late', and 'planned\_early' for alternate ways to test for ACCEPTANCE\_NOT\_PLANNED, ACCEPTANCE\_PLANNED\_LATE, and ACCEPTANCE\_PLANNED\_EARLY Problems, respectively.

Note well: the lack of a Problem here only indicates that a plan to meet this Acceptance is being attempted. There may be feasibility Problems with the plans associated with fulfilling this Acceptance in addition to any Problem here.

Properties: obsolete=True Export-Only Field

planned\_quantity\_problem - - a Problem field of model Item\_Acceptance  
 Obsolete! Use Item\_Acceptance 'problems'. The ACCEPTANCE\_PLANNED\_QUANTITY Problem, if any, that exists between this Item\_Acceptance and the Operation\_Plan currently planned to satisfy it ('delivery\_plan'). See fields 'not\_planned', 'planned\_short', and 'planned\_excess' for more convenient ways to test for ACCEPTANCE\_NOT\_PLANNED, ACCEPTANCE\_PLANNED\_SHORT, and ACCEPTANCE\_PLANNED\_EXCESS Problems, respectively.

Models	Item_Acceptance Model
--------	-----------------------

Note well: the lack of a Problem here only indicates that a plan to meet this Acceptance is being attempted. There may be feasibility Problems with the plans associated with fulfilling this Acceptance in addition to any Problem here.

Properties: obsolete=True Export-Only Field

owner - - a Delivery\_Acceptance field of model Item\_Acceptance

Properties: Export-Only Field

5.1.3.2 Seller\_Plan Model

Seller\_Plan -- a submodel of model Plan

The master plan for a Seller. It includes the Seller's Forecast of how much of each of its Products could be sold, the sales the Seller has committed to making, the allocations made from the supplying Sites to the Seller, and the customer Promises the Seller has made based on those allocations.

The Seller\_Plan model has these submodels :  
Forecast.

The Seller\_Plan model has fields that references these models :  
Seller, Seller\_Plan, Forecast.

These models have a field that is a Seller\_Plan model :  
Seller, Plan, Forecast, Seller\_Plan, Request, Delivery\_Request.

The key field for this model is seller  
This model may be extended with user-defined fields.

seller -- a Seller field of model Seller\_Plan  
The Seller that this Seller\_Plan is planning.

This field is not really settable (to be fixed).  
Default: None -- this is a key field

organization -- a Seller\_Plan field of model Seller\_Plan  
The Seller\_Plan for this seller's organization.  
Properties: Export-Only Field

members -- a List(Seller\_Plan) field of model Seller\_Plan  
This List contains the Seller\_Plan for each of this seller's immediate members. Use 'seller.recurse(# members)' to get the entire tree below 'seller'.  
Properties: Export-Only Field

forecasts -- a list of Forecast submodels of model Seller\_Plan  
The forecasts and allocations (the master plans) for each Product and Product\_Group of this Seller and its organizations (e.g., 'all\_products' + 'all\_product\_groups'). These Forecasts form a hierarchy that parallels the Seller's Product\_Group hierarchy. The set of INDIVIDUAL Forecasts constitutes the master plan for the Seller. The GROUP Forecasts are essentially a tool for aggregating and disaggregating the INDIVIDUAL Forecasts.

active\_forecasts -- a List(Forecast) field of model Seller\_Plan  
The 'active' forecasts for this Seller Plan (see Forecast.active). The returned list will include all GROUP forecasts.  
Properties: Export-Only Field

forecast (Symbol) -- a Forecast field of model Seller\_Plan  
The Forecast of this Seller\_Plan for the Product or Product\_Group with the specified name. That Product or Product\_Group may be defined by this Seller, or one of its organizations.

Note that 'forecast(name)' is equivalent to 'forecast.find(name)', but is simpler, clearer, and more efficient. Also note that if a Product and a Product\_Group have the same name, this will return the Forecast for the Product\_Group. Use 'product\_forecast(name)' to get the Product's Forecast.  
Properties: Export-Only Field

top\_forecasts -- a List(Forecast) field of model Seller\_Plan  
The Forecasts for the Seller's top\_products and top\_product\_groups, those that are the top of a Product\_Group hierarchy. These are the Forecasts that do not appear within any other GROUP Forecast. The (unspecified) Forecast is not included.  
Properties: Export-Only Field

active\_top\_forecasts -- a List(Forecast) field of model Seller\_Plan  
The active Forecasts for the Seller's top\_products and top\_product\_groups.  
Properties: Export-Only Field

product\_root\_forecasts -- a List(Forecast) field of model Seller\_Plan  
The INDIVIDUAL Product\_Root Forecasts (Forecasts for which this Seller defines the root). The (unspecified) Forecast is not included.  
Properties: Export-Only Field

active\_product\_root\_forecasts -- a List(Forecast) field of model Seller\_Plan  
The active INDIVIDUAL Product\_Root Forecasts.



Models	Seller_Plan Model
--------	-------------------

Properties:    Export-Only Field

**product\_forecasts** - - *a List(Forecast) field of model Seller\_Plan*

The INDIVIDUAL Product Forecasts. The [unspecified] Forecast is not included.

Properties:    Export-Only Field

**active\_product\_forecasts** - - *a List(Forecast) field of model Seller\_Plan*

The active INDIVIDUAL Product Forecasts.

Properties:    Export-Only Field

**product\_forecast (Symbol)** - - *a Forecast field of model Seller\_Plan*

The INDIVIDUAL Product Forecast of this Seller\_Plan for the Product with the specified name. That Product may be defined by this Seller, or one of its organizations.

Note that 'product\_forecast(name)' is equivalent to 'product\_forecasts.find(name)'; but is simpler, clearer, and more efficient.

Properties:    Export-Only Field

**forecast\_horizon** - - *a List(Date\_Range) field of model Seller\_Plan*

The "forecasting buckets". The List of Date\_Ranges that define the forecast horizon, that define the Date\_Range of each Forecast\_Entry. This is defined by the 'forecast\_horizon' extension of the Seller.

Note that a horizon will always consist of a consecutive series of adjacent, but non-overlapping, Date\_Ranges that cover the full planning horizon.

Properties:    Export-Only Field

**atp\_horizon** - - *a List(Date\_Range) field of model Seller\_Plan*

The "allocation buckets" or "ATP buckets". The List of Date\_Ranges that define the atp horizon, that define the Date\_Range of each ATP\_Entry within a Forecast for this Seller\_Plan. This is defined by the 'atp\_horizon' extension of this Seller\_Plan's Seller.

Note that a horizon will always consist of a consecutive series of adjacent, but non-overlapping, Date\_Ranges that cover the full planning horizon.

Properties: ✓    Export-Only Field

**actual\_requests** - - *a List(Request) field of model Seller\_Plan*

The customer Requests that have been placed through this Seller.

Properties:    Export-Only Field

Models	Seller_Plan Model
--------	-------------------

**actual\_promises** - - *a List(Promise) field of model Seller\_Plan*

The customer Promises that have been made through this Seller.

Properties:    Export-Only Field

**problems, problems (Date\_Range) , problems (Problem\_Category) , problems (Date\_Range, Problem\_Category)** - - *a List(Problem) field of model Seller\_Plan*

The SELLER Problems associated with the Forecast\_Entries of this Seller\_Plan.

Properties:    Export-Only Field

**accept\_as\_allocated** - - *a Void field of model Seller\_Plan*

Accept the allocated values in all of the Forecast\_Entry's of all the 'Forecasts' in this Seller\_Plan and its members. This does 'accept\_as\_allocated' for each of the 'Forecasts', which in turn does 'accept\_as\_allocated' for each of its entries, which copies the 'allocated' and 'retain\_from\_allocated' fields to the 'accepted' and 'retain\_from\_accepted' fields in that Forecast\_Entry and in the corresponding Forecast\_Entry(s) in every Forecast in the specific and member trees below it. It also sets 'date\_accepted' in all of those Forecast\_Entry's to 'now'.

Note that the full hierarchy is done because it is in general a bad idea to mix one set of allocations with another -- they may be completely inconsistent. However, you can always set any or all of the fields directly.

Properties:    command=True    Export-Only Field

**owner** - - *a Plan field of model Seller\_Plan*

Properties:    Export-Only Field

5.1.3.2.1 Forecast Model

Forecast -- a submodel of model Seller\_Plan

In each Seller\_Plan there is one Forecast for each Product and for each Product\_Group defined by that Seller or its organizations. Each Forecast is made up of a series of Forecast\_Entry's which each contain a raw 'forecasted' value, a 'committed' forecast value, an 'allocated' value, a 'consumed' value, and thus ATP values. There is one Forecast\_Entry for each time period defined by the Seller's 'forecast\_horizon'.

Forecasts are effectively organized into two hierarchies: the Seller hierarchy as defined by the Seller 'organization' field and the Product\_Group hierarchy defined by the 'product\_groups' of the Seller. Since the Product\_Group hierarchy is defined for each Seller, the Seller hierarchy is in a sense the "outer" hierarchy.

The 'level' extension of Forecast indicates whether it is an INDIVIDUAL Forecast (for a Product) or a GROUP Forecast (for a Product\_Group). Note that the set of INDIVIDUAL Forecasts constitute the master plan for the Seller. The GROUP Forecasts are essentially a tool for aggregating and disaggregating the INDIVIDUAL Forecasts. Such aggregation is the essence of intelligent forecast management, since forecasts are much more accurate in aggregate.

However, the GROUP Forecasts are not limited to aggregation. Selling a GROUP Forecast\_Entry's 'forecasted' or 'committed' fields to a new value will set the same field of each of its 'sub\_forecasts' such that the sum of those 'sub\_forecasts' equals the value set there. The property that the GROUP Forecast's values are the sums of its 'sub\_forecasts' is always maintained. In that sense, a GROUP Forecast is just a tool for seeing the aggregated total and for adjusting a group of Forecast values evenly.

Selling a Forecast\_Entry field immediately propagates the change up to any GROUP Forecasts that contain it, and immediately disaggregates the change down to any 'sub\_forecasts'. How the change is disaggregated depends upon the value of the 'use\_sid\_split' field. If "false", then the change is split out in the same proportions that the 'sub\_forecasts' currently have (thereby not changing the percentage splits). If "true", then the change is split according to the 'sid\_split's in the Product definitions, regardless of the current distribution.

The Product\_Group hierarchies are a tool for a single Seller to manipulate the Forecasts that he or she owns. As such, the 'forecasted' and 'committed' values in the Product\_Group hierarchy are tied directly to one another -- changes are immediately propagated throughout the hierarchies.

In contrast, the Seller hierarchy is about coordination of multiple independent Sellers, who may each have responsibility, and therefore control, over their own forecasts, commitments, and allocations. In this case, the 'forecasted' and 'committed' fields are not tied directly to one another. Rather, an organization has 'member\_forecasted' and 'member\_committed' fields which are the sums of its 'member' Seller's values. These sums are independent of (though related and compared to) its own values. As such, disaggregation is not performed for 'forecasted' and 'committed' values.

Disaggregation through the Seller hierarchy is performed for the 'allocated' field. It is done according to the 'allocation\_policy' in the Allocation\_Policy of this Seller for this Product.

The model has selectors:  
level.

The Forecast model has these submodels :  
Forecast\_Entry.

The Forecast model has fields that references these models :  
Forecast, Request, Forecast\_Entry, Seller\_Plan.

These models have a field that is a Forecast model :  
Plan, Forecast, Seller\_Plan, Request, Forecast\_Entry, ATP\_Entry, Item\_Promise, Product\_Available\_To\_Promise.

The key field for this model is name  
This model may be extended with user-defined fields.

level -- an extension selector of model Forecast  
If "GROUP", then this Forecast is for a Product\_Group, an aggregation of several Products' Forecasts. The Product\_Group defines how this Forecast is propagated down to 'sub\_groups' or 'products'.

If "INDIVIDUAL", then this Forecast is for an individual Product. The Product defines how this Forecast is converted into Requests upon this Site.

Default: INDIVIDUAL  
Properties: Export-Only Field  
Extensions:  
INDIVIDUAL, GROUP.

*name* - - *a Symbol field of model Forecast*

If 'level' is "GROUP", then this field is 'product\_group.name', otherwise it is 'product.name'. This field is not settable.

Default: None -- this is a key field

Properties: Export-Only Field

*remark* - - *a String field of model Forecast*

Any remarks about this Forecast.

Default: none

*active* - - *a Logical field of model Forecast*

A Forecast is 'active' if any of its fields has been set to other than the default value.

The conversion to "active" can occur either through directly setting a specific forecast or by setting a higher level GROUP Forecast. GROUP Forecasts will always be inactive since setting a field in a GROUP Forecast simply sets the fields of the members of the group.

The default for each of the Forecast model fields which return List(Forecast) is to return the complete set of Forecasts (both 'inactive' and 'active'). Each such field will also have an 'active' counterpart. For example, Forecast.member\_forecasts() returns all Forecasts of the member Sellers, but Forecasts.active\_member\_forecasts() returns only those Forecasts actively in use by the member Sellers.

The 'active' vs 'inactive' distinction is useful in situations where the number of possible Seller/Product Forecast combinations is much larger than the number of Seller/Product combinations actively being forecasted.

The 'active' field is not settable.

Properties: Export-Only Field

*root* - - *a Forecast field of model Forecast*

The root Forecast in the Product-Seller tree. If this is a GROUP Forecast, then this returns the Forecast for the 'product\_group.root'. Similarly, if this is an INDIVIDUAL Forecast, then this returns the Forecast for the Product in the Seller that defined the Product\_Root.

Properties: Export-Only Field

*member\_forecasts* - - *a List(Forecast) field of model Forecast*

A List of the Forecasts of this Forecast's Seller's 'members'.

Properties: Export-Only Field

*active\_member\_forecasts* - - *a List(Forecast) field of model Forecast*

A List of the 'active' Forecasts of this Forecast's Seller's 'members'.

Properties: Export-Only Field

*generic\_forecasts* - - *a List(Forecast) field of model Forecast*

A List of the Forecasts that correspond to this Product's 'generic\_products'. These are the Forecasts that represent the more generic demand being forecasted and allocated. This Forecast is essentially a subset of those more generic Forecasts. This Forecast will appear in these Forecast's 'specific\_forecasts'.

Properties: Export-Only Field

*active\_generic\_forecasts* - - *a List(Forecast) field of model Forecast*

A List of the 'active' Forecasts that correspond to this Product's 'generic\_products'.

Properties: Export-Only Field

*specific\_forecasts* - - *a List(Forecast) field of model Forecast*

A List of the Forecasts that correspond to this Product's 'specific\_products'. These are the Forecasts that represent the more specific demand being forecasted and allocated. This Forecast is generically covering all of the 'specific\_products', and thus will be at least the sum of those. See the various fields of Forecast\_Entry prefixed with 'specifics\_'. This Forecast will appear in these Forecast's 'generic\_forecasts'.

Properties: Export-Only Field

*active\_specific\_forecasts* - - *a List(Forecast) field of model Forecast*

A List of the "active" Forecasts that correspond to this Product's 'specific\_products'.

Properties: Export-Only Field

*request* - - *a Request field of model Forecast*

The Forecast Request corresponding to this Forecast. If there is none, create one.

There is no Forecast Request for either Group or inactive Forecasts

Properties: Export-Only Field

*entries* - - *a list of Forecast\_Entry submodels of model Forecast*

In each Forecast, there is one Forecast\_Entry for each Date\_Range in the Seller\_Plan's 'forecast\_horizon'. Each Forecast\_Entry has the Date\_Range it is forecasting, a raw 'forecasted' value, a 'committed' forecast value, an 'allocated' value that it can promise, a 'consumed' value that it has promised, and thus an 'gap' value that is still available to promise. It also contains fields that aggregate up those same values from this Seller's 'members', such as 'members\_forecasted', 'members\_committed', and 'members\_allocated'.

Models	Forecast Model
--------	----------------

**entries\_consummed** - - *a List(Quantity) field of model Forecast*  
 The total consumed quantity for each of the entries. (See Forecast\_Entry.consumed)  
 Properties:   Export-Only Field

**entries\_members\_consummed** - - *a List(Quantity) field of model Forecast*  
 The total Quantity of this Product for which actual Promises have been made by the 'members' of this Seller for each of the entries. (See Forecast\_Entry.members\_consumed)  
 Properties:   Export-Only Field

**accept\_as\_allocated** - - *a Void field of model Forecast*  
 Accept the allocated values in all of the Forecast\_Entry's of this Forecast and then 'accept\_as\_allocated' any specific 'forecasts' and any 'member\_forecasts'. This does 'accept\_as\_allocated' for each of the 'entries', which copies the 'allocated' and 'retain\_from\_allocated' fields to the 'accepted' and 'retain\_from\_accepted' fields in that Forecast\_Entry and in the corresponding Forecast\_Entry(s) in every Forecast in the specific and member trees below it. It also sets 'date\_accepted' in all of those Forecast\_Entry's to 'now'.

Note that the full hierarchy is done because it is in general a bad idea to mix one set of allocations with another - - they may be completely inconsistent. However, you can always set any or all of the fields directly.  
 Properties:   command=True   Export-Only Field

**allocate\_allocated\_available** - - *a Void field of model Forecast*  
 This command allocates any quantity in 'allocated\_available' of each Forecast\_Entry to the Forecast\_Entry's of member sellers  
 Properties:   command=True   Export-Only Field

**owner** - - *a Seller\_Plan field of model Forecast*  
 Properties:   Export-Only Field

**product** - - *a Product field of model INDIVIDUAL*  
 The Product for which demand this forecasts.  
 Properties:   standard=True   Export-Only Field

**product\_group** - - *a Product\_Group field of model GROUP*  
 The Product for which demand this forecasts.  
 Properties:   standard=True   Export-Only Field

Models	Forecast Model
--------	----------------

**sub\_forecasts** - - *a List(Forecast) field of model GROUP*  
 The Forecasts for the 'sub\_groups' and 'products' of this Forecast's 'product\_group'.  
 Properties:   standard=True   Export-Only Field

**active\_sub\_forecasts** - - *a List(Forecast) field of model GROUP*  
 The Forecasts for the 'sub\_groups' and active 'products' of this Forecast's 'product\_group'.  
 Properties:   standard=True   Export-Only Field

**active\_leaf\_product\_forecasts** - - *a List(Forecast) field of model GROUP*  
 The Forecasts for the active 'products' of this Forecast's 'product\_group'. Returns all active descendants of this GROUP Forecast that are INDIVIDUAL forecasts. In other terms, these are the active "leaf" Forecasts of this product sub-hierarchy.  
 Properties:   standard=True   Export-Only Field

5.1.3.2.1.1 Standard Extensions of model Forecast  
5.1.3.2.1.1.1 level extensions of model Forecast  
5.1.3.2.1.1.1 INDIVIDUAL -- a level extension of model Forecast

An INDIVIDUAL Forecast is for an individual Product. The 'product' defines how this Forecast is converted into Requests upon this Site.

The INDIVIDUAL model has these submodels :  
ATP\_Entry.

The INDIVIDUAL model has fields that references these models :  
Product, ATP\_Entry.

product - - a Product field of model INDIVIDUAL  
The Product for which demand this forecasts.  
Properties: standard=True Export-Only Field

atp\_entries - - a list of ATP\_Entry submodels of model INDIVIDUAL  
In each Forecast, there is one ATP\_Entry for each Date\_Range in the Seller\_Plan's list of Date\_Ranges. 'atp\_horizon': Each ATP\_Entry has the Date\_Range for which it maintains allocations, an 'allocated' value that it can promise, a 'consumed' value that maintains the amount already promised, and an 'atp' value that is still available to promise.

5.1.3.2.1.1.2  
GROUP -- a level extension of model Forecast

A GROUP Forecast is for a Product\_Group, an aggregation of several Products' Forecasts. The 'product\_group' defines how this Forecast is propagated down to the Product\_Group's 'sub\_groups' and 'products' :

The GROUP model has fields that references these models :  
Product\_Group.

product\_group - - a Product\_Group field of model GROUP  
The Product for which demand this forecasts.  
Properties: standard=True Export-Only Field

sub\_forecasts - - a List(Forecast) field of model GROUP  
The Forecasts for the 'sub\_groups' and 'products' of this Forecast's 'product\_group'.  
Properties: standard=True Export-Only Field

active\_sub\_forecasts - - a List(Forecast) field of model GROUP  
The Forecasts for the 'sub\_groups' and active 'products' of this Forecast's 'product\_group'.  
Properties: standard=True Export-Only Field

active\_leaf\_product\_forecasts - - a List(Forecast) field of model GROUP  
The Forecasts for the active 'products' of this Forecast's 'product\_group'. Returns all active descendants of this GROUP Forecast that are INDIVIDUAL forecasts. In other terms, these are the active "leaf" Forecasts of this product sub-hierarchy.  
Properties: standard=True Export-Only Field

use\_split - - a Logical field of model GROUP  
This field is obsolete. You must now set the 'use\_split' field in the Product\_Group. Note that setting the field at the Product\_Group is not exactly equivalent. The split now applies to all Forecasts of the Product\_Group.  
Properties: obsolete=True Export-Only Field

Models	Forecast_Entry Model
--------	----------------------

### 5.1.3.2.1.2 Forecast\_Entry Model

Forecast\_Entry -- *a submodel of model Forecast*

In each Forecast, there is one Forecast\_Entry for each Date\_Range in the Seller\_Plan's forecast\_horizon. Each Forecast\_Entry has the Date\_Range it is forecasting, a raw 'forecasted' value, a 'committed' forecast value, an 'allocated' value that it can promise, a 'consumed' value that it has promised, and thus an 'atp' value that is still available to promise. It also contains fields that aggregate up those same values from this Seller's 'members', such as 'members\_forecasted', 'members\_committed', and 'members\_allocated'.

The model has selectors:  
forecast\_policy, allocation\_policy.

The Forecast\_Entry model has fields that references these models :  
Forecast.

These models have a field that is a Forecast\_Entry model :  
Forecast, Delivery\_Request, Product\_Available\_To\_Promise, NEGATIVE\_ATP, NEGATIVE\_PLANNED\_ATP, OVER\_COMMITTED, OVER\_CONSUMED, UNALLOCATED\_FORECAST.

The key field for this model is delivery\_dates  
This model may be extended with user-defined fields.

delivery\_dates -- *a Date\_Range field of model Forecast\_Entry*  
The delivery Dates forecasted by this Forecast\_Entry. This cannot be set -- it is determined by the Seller\_Plan's 'forecast\_horizon'. All forecasting must be done for the same Date\_Ranges for them to be comparable and aggregatable.

This field is not really settable (to be read-only soon).  
Default: None -- this is a key field  
Properties: Export-Only Field

date\_forecasted -- *a Date field of model Forecast\_Entry*  
The Date that the 'forecasted' value was last set. Setting 'forecasted' will set 'date\_forecasted' to 'max(date\_forecasted, now)'. As 'date\_forecasted' is only for informational purposes, no problems are detected relative to it.  
Default: the Date when 'forecasted' was set (initially infinite past)

Models	Forecast_Entry Model
--------	----------------------

forecasted -- *a Quantity field of model Forecast\_Entry*  
The Quantity of this 'product' or 'product\_group' that the Seller believes can be sold for these 'delivery\_dates'. This is the market potential. This may be an aggressive forecast -- it is not a commitment (see 'committed'). Rather, it is an upper bound on what can be 'committed'. Setting 'committed' higher than 'forecasted' results in an OVER\_COMMITTED Problem.

Note that the default, "oo", implies that the market can support any level of sales. Thus, any 'committed' Forecast is reasonable. This makes this field essentially irrelevant -- the market potential is not a limit. By default, then, users can ignore this field and just set the 'committed' Forecast values.

If 'override\_members\_forecasted' is "true", then changes in 'members\_forecasted' will have no effect on this 'forecasted' Quantity. This allows an 'organization' forecast to override what the 'members' are forecasting. For example, the 'members\_forecasted' may be 600, but this Seller may believe that the total that will be sold during this period will be 800, independent of what the 'members' estimate. If the 'members' change their estimates to 700, this Seller still wants his 'forecasted' to remain 800.

In contrast, if 'override\_members\_forecasted' is "false", then any increase or decrease in 'members\_forecasted' will increase or decrease this 'forecasted' Quantity by the same Quantity. This allows an 'organization' to add a certain Quantity either for its own sales or as an estimate of how much the 'members' typically under or over forecast. Thus, in the previous example, when this Seller set 'forecasted' to 800, the Seller may have meant "I want my 'forecasted' to be 200 more than my 'members' estimate; so, when the 'members' sum increases to 700, this 'forecasted' should rise to 900 (remaining 200 more)".

Note that 'forecasted' cannot be set to less than 'specifics\_forecasted', though typically 'forecasted' is significantly more than 'specifics\_forecasted' anyway. If this Forecast has 'specific\_forecasts' (if its 'product' is the Generic\_Product of another Product), then this 'forecasted' Quantity includes all of the 'forecasted' Quantities of its 'specific\_forecasts'. Thus, if 'specifics\_forecasted' is ever increased to larger than 'forecasted', then 'forecasted' will be increased to match.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.  
Default: oo

cumulative\_forecasted -- *a Quantity field of model Forecast\_Entry*  
The sum of 'forecasted' for each Forecast\_Entry from the first through this one.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.

Properties: Export-Only Field

**specifics\_forecasted** - - *a Quantity field of model Forecast\_Entry*  
The sum of the 'forecasted' quantities for these 'delivery\_dates' for the 'specific\_forecasts' (the 'forecasts' for this Product's 'specific\_products'). This will be zero if this is a GROUP Forecast or if this Product is not a Generic\_Product (if it has no 'specific\_products').

Note that 'forecasted' cannot be set to less than 'specifics\_forecasted', and that the 'forecasted' value will be automatically increased such that it is always at least as much as 'specifics\_forecasted', since the 'specific\_forecasts' are essentially the subsets of the generic demand modelled by this Forecast. Typically, however, 'forecasted' is more than 'specifics\_forecasted' due to the uncertainty in the demand for specific products.

This Quantity is converted to the 'unit' of the product.  
Properties: Export-Only Field

**members\_forecasted** - - *a Quantity field of model Forecast\_Entry*  
The sum of the 'forecasted' quantities for these 'delivery\_dates' for this Product by the 'member\_forecasts' (the 'forecasts' of the Sellers that are members of this Seller). This will be zero if this Seller has no 'members'.

If 'forecasted' is less (more) than this, then the Seller is asserting that the 'members' are overly optimistic (pessimistic).

If 'override\_members\_forecasted' is "true", then changes in 'members\_forecasted' will have no effect on this 'forecasted' Quantity. In contrast, if 'override\_members\_forecasted' is "false", then any increase or decrease in 'members\_forecasted' will increase or decrease this 'forecasted' Quantity by the same Quantity. See the further discussion and examples discussed in the 'forecasted' field.

This field can be set, which results in proportional increase or decrease of all of the corresponding 'forecasted' fields of the 'member\_forecasts' which make up this sum. So, consider the case that there are four 'member\_forecasts' with corresponding 'forecasted' quantities of 300, 200, 100, and 0, making this field 600; if you set this field to 300, then the 'forecasted' fields of the 'member\_forecasts' will be set to 150, 100, 50, and 0, respectively. If this field is set to 0, then 'forecasted' fields of the 'member\_forecasts' will be set to 0. Subsequently, if this value is set to 600, this value will be equally distributed among the members, i.e. the 'forecasted' fields of the all 'member\_forecasts' will be set to 150.

This Quantity is converted to the 'unit' of the product or 'product\_group'.

Default: sum of 'forecasted' for all 'members'

**override\_members\_forecasted** - - *a Logical field of model Forecast\_Entry*  
If 'override\_members\_forecasted' is "true", then changes in 'members\_forecasted' will have no effect on this 'forecasted' Quantity. In contrast, if 'override\_members\_forecasted' is "false", then any increase or decrease in 'members\_forecasted' will increase or decrease this 'forecasted' Quantity by the same Quantity. See the further discussion and examples discussed in the 'forecasted' field.

If this is an INDIVIDUAL Forecast, then if 'prod-uc1.always\_override\_members\_forecasted' is "true", then this value is "true" (and cannot be set to "false"). If this is a GROUP Forecast, then this value is "true" (and cannot be set to "false"). Otherwise, this field may be set either "true" or "false".

Currently, this field is not settable; you can only set 'prod-uc1.always\_override\_members\_forecasted'.  
Properties: Export-Only Field

**date\_committed** - - *a Date field of model Forecast\_Entry*  
The Date that the 'committed' value was last set (requested). Selling 'committed' will set 'date\_committed' to 'max(date\_committed, now)'. If 'date\_committed' is later than 'date\_allocated', then an UNALLOCATED\_FORECAST Problem will be created to indicate that a different committed forecast has been created but not yet satisfied or rejected.  
Default: the Date when 'committed' was set (initially infinite past)

**committed** - - *a Quantity field of model Forecast\_Entry*  
The Quantity of this product or 'product\_group' that this Seller is willing to commit to selling for these 'delivery\_dates'. This could also be called "requested allocation". It is the Quantity that will be allocated to this particular Seller if there is sufficient supply.

Typically, the Seller organization will set this higher than the 'members\_committed', committing that several of the 'members' will sell more than they committed (though which of the 'members' will do so may be anyone's guess). In this way, additional available-to-promise will be 'allocated' to the organization, available to the 'members' on a first-come-first-served basis. So, as members sell more than they 'committed', and thus lack 'allocated' available-to-promise, they can begin to consume from their organization's ATP.

Note that 'committed' is not synonymous with ATP. The 'committed' value results in forecast Requests. If those Requests can be met and Promises are made from the supplier Sites, then those Promises constitute the allocated Quantity, the unconsumed portion of which is ATP. Those Promises have been allocated to the Seller, and the Seller can use those Promises to immediately make Promises to actual Requests from customer Sites.

If this is set larger than 'forecasted' (the estimated market demand), an OVER\_COMMITTED Problem will be raised. If a separate 'forecasted' value is not desired, set all 'forecasted' values to "00" (the default) which will effectively make those limits vanish.

If 'override\_members\_committed' is "true", then changes in 'members\_committed' will have no effect on this 'committed' Quantity. This allows an 'organization' commitment to override what the 'members' are committing. For example, the 'members\_committed' may be 600, but this Seller may believe that the total that can be sold during this period will be 800, independent of what the 'members' estimate. If the 'members' change their estimates to 700, this Seller still wants his 'committed' to remain 800.

In contrast, if 'override\_members\_committed' is "false", then any increase or decrease in 'members\_committed' will increase or decrease this 'committed' Quantity by the same Quantity. This allows an 'organization' to add a certain Quantity either for its own sales or as an estimate of how much the 'members' typically under or over forecast. Thus, in the previous example, when this Seller set 'committed' to 800, the Seller may have meant "I want my 'committed' to be 200 more than my 'members' commitment; so, when the 'members' sum increases to 700, this 'committed' should rise to 900 (remaining 200 more).

Note that 'committed' cannot be set to less than 'specifics\_committed', though typically 'committed' is significantly more than 'specifics\_committed' anyway. If this Forecast has 'specific\_forecasts' (if its 'product' is the Generic\_Product of another Product), then this 'committed' Quantity includes all of the 'committed' Quantities of its 'specific\_forecasts'. Thus, if 'specifics\_committed' is ever increased to larger than 'committed', then 'committed' will be increased to match.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.  
Default: 0

**cumulative\_committed** - - a Quantity field of model Forecast\_Entry  
The sum of 'committed' for each Forecast\_Entry from the first through this one.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.  
Properties: Export-Only field

**specifics\_committed** - - a Quantity field of model Forecast\_Entry  
The sum of the 'committed' quantities for these 'delivery\_dates' for the 'specific\_forecasts' (the Forecasts for this Product's 'specific\_products'). This will be zero if this is a GROUP Forecast or if this Product is not a Generic\_Product (if it has no 'specific\_products').

Note that 'committed' cannot be set to less than 'specific\_committed', and that the 'committed' value will be automatically increased such that it is always at least as much as 'specifics\_committed', since the 'specific\_forecasts' are essentially the subsets of the generic demand modelled by this Forecast. Typically, however, 'committed' is more than 'specifics\_committed' due to the uncertainty in the demand for specific products.

Further, note that the forecast requests generated by this Forecast are only for the delta between 'committed' and 'specifics\_committed'. The 'specifics\_committed' Quantity is covered by the forecast requests generated by the 'specific\_forecasts'. Of course, also subtracted out is the Quantity for any actual Requests or Promises that are planned, which are typically all on the 'specific\_forecasts'.

This Quantity is converted to the 'unit' of the 'product'.  
Properties: Export-Only field

**members\_committed** - - a Quantity field of model Forecast\_Entry  
The sum of the 'committed' quantities for these 'delivery\_dates' for this Product by the 'member\_forecasts' (the Forecasts of the Sellers that are 'members' of this Seller). This will be zero if this Seller has no 'members'.

If 'committed' is less (more) than this, then the Seller is asserting that the 'members' are overly optimistic (pessimistic).

If 'override\_members\_committed' is "true", then changes in 'members\_committed' will have no effect on this 'committed' Quantity. In contrast, if 'override\_members\_committed' is "false", then any increase or decrease in 'members\_committed' will increase or decrease this 'committed' Quantity by the same Quantity. See the further discussion and examples discussed in the 'committed' field.



This field can be set, which results in proportional increase or decrease of all of the corresponding 'committed' fields of the 'member\_forecasts' which make up this sum. So, consider the case that there are four 'member\_forecasts' with corresponding 'committed' quantities of 300, 200, 100, and 0, making this field 600; if you set this field to 300, then the 'committed' fields of the 'member\_forecasts' will be set to 150, 100, 50, and 0, respectively. If this field is set to 0, then 'committed' fields of the 'member\_forecasts' will be set to 0. Subsequently, if this value is set to 600, this value will be equally distributed among the members, i.e. the 'committed' fields of the all 'member\_forecasts' will be set to 150.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.

Default: sum of 'committed' for all 'members'

**override\_members\_committed** - - *a Logical field of model Forecast\_Entry*

If 'override\_members\_committed' is "true", then changes in 'members\_committed' will have no effect on this 'committed' Quantity. In contrast, if 'override\_members\_committed' is "false", then any increase or decrease in 'members\_committed' will increase or decrease this 'committed' Quantity by the same Quantity. See the further discussion and examples discussed in the 'committed' field.

If this is an **INDIVIDUAL** Forecast, then if 'prod-

uct.always\_override\_members\_committed' is "true", then this value is "true" (and cannot be set to "false"). If this is a **GROUP** Forecast, then this value is "true" (and cannot be set to "false"). Otherwise, this field may be set either "true" or "false".

Currently, this field is not settable; you can only set 'prod-

uct.always\_override\_members\_committed'.

Properties: Export-Only Field

**retain\_from\_allocated** - - *a Quantity field of model Forecast\_Entry*

The Quantity of the 'allocated' (and 'planned') that should be retained at this Seller.

This Quantity is 'available' from this Seller, but will not be allocated down to the members not available for use by the members first-come-first-served.

This field, though computed by the 'allocation\_policy' and related fields of the Product, is settable. Setting 'retain' will also set 'lock\_retain\_from\_allocated' to "true" -- which will prevent this field from being recomputed by the 'allocation\_policy' of the Product -- it will remain at the Quantity set into it.

This allows the user to easily manipulate the 'retain\_from\_allocated' value for a particular bucket, without needing to modify the Product parameters. This is often used to "release" some of the retained Quantity when more is needed by the 'members'.

Default: computed from the Product fields

**lock\_retain\_from\_allocated** - - *a Logical field of model Forecast\_Entry*

If "false", then the 'retain' Quantity is computed from the Product 'allocation\_policy' and related fields. If "true", then the current 'retain' Quantity will be used, overriding the setting that would normally be computed based upon the fields in the Product.

This allows the user to easily manipulate the 'retain' value for a particular bucket, without needing to modify the Product parameters. This is often used to "release" some of the 'retained' Quantity when more is needed by the 'members'.

Note that setting this to "false" will cause the 'retain' Quantity, and therefore the 'retained' Quantity, to be recomputed from the Product fields.

Default: false

**retain\_from\_accepted** - - *a Quantity field of model Forecast\_Entry*

The Quantity of the 'accepted' that should be retained at this Seller. This Quantity is 'available' from this Seller, but will not be available to the 'members' (as first-come-first-served ATP).

This field is set from 'retain\_from\_allocated' when 'accept\_as\_allocated' is called. This field can also be set directly. This allows the user to easily manipulate the retained value for a particular bucket, which is often used to "release" some of the retained Quantity when more is needed by the 'members'.

Default: 0

**planned** - - *a Quantity field of model Forecast\_Entry*

The Quantity of this 'product' or 'product\_group' that is currently planned to be delivered. This is an "unpromised" or "what-if" variation of 'allocated'. It is allocated to the 'members' using the same 'allocation\_policy' as 'allocated'. If current plans are not satisfying the Promises that make up 'allocated', then this value may be less than 'allocated'. On the other hand, if a planning effort is underway to increase 'allocated', this value may be larger than 'allocated'. In a sense, this often represents what 'allocated' will be soon -- the "what if I planned like this, what would be my 'allocated' value?"

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.

Properties: Export-Only Field

**cumulative\_planned** - - *a Quantity field of model Forecast\_Entry*

The sum of 'planned' for each Forecast\_Entry from the first through this one.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.

Properties: Export-Only Field

**specifics\_planned** - - *a Quantity field of model Forecast\_Entry*  
The sum of the planned quantities for these *delivery\_dates* for the *specific\_forecasts* (the *Forecast* for this *Product's specifics\_products*). This will be zero if this is a *GROUP Forecast* or if this *Product* is not a *Generic\_Product* (if it has no *specific\_products*).

Note that *planned* will never be less than *specifics\_planned*, since the *specific\_forecasts* are essentially the subsets of the generic demand modelled by this *Forecast*. Typically, however, *planned* is more than *specifics\_planned* due to the uncertainty in the demand for specific products.

This Quantity is converted to the unit of the product:

Properties: Export-Only Field

**members\_planned** - - *a Quantity field of model Forecast\_Entry*  
The sum of the planned quantities for these *delivery\_dates* for this *Product* by the *member\_forecasts* (the *Forecasts* of the *Sellers* that are *members* of this *Seller*). This will be zero if this *Seller* has no *members*.

This Quantity is converted to the unit of the product or product\_group:

Properties: Export-Only Field

**date\_allocated** - - *a Date field of model Forecast\_Entry*  
The Date that the allocated value was last set (promised). Setting *allocated* will set *date\_allocated* to *max(date\_allocated, now)*.

If *date\_committed* is later than *date\_allocated*, then a *FORECAST\_NOT\_ALLOCATED* (an *UNALLOCATED\_FORECAST*) Problem will be created to indicate that a different commitment has been created but not yet satisfied or rejected. Rejecting it is as simple as setting this field to *now*.

If *date\_allocated* is later than *date\_accepted*, then an *ALLOCATION\_NOT\_ACCEPTED* Problem will be created to indicate that a different allocation has been made, but has not yet been accepted or rejected. Rejecting it is as simple as setting *date\_accepted* to *now*. Accepting it can be done by *accept\_as\_allocated*.  
Default: the Date when *allocated* was set (initially infinite past)

**allocated** - - *a Quantity field of model Forecast\_Entry*  
The Quantity of this product or product\_group for which forecast Promises have been allocated to the top-seller for these *available\_dates*. This could also be called "preliminary gross ATP" -- the ATP prior to being consumed by actual Promises. The ATP is preliminary since this *allocated* Quantity may not yet be accepted. The *allocated\_available* is this *allocated* minus the *consumed* (the actual Promises that consume from this *Forecast\_Entry*).

If *allocated* is less than *committed*, then additional allocation may be obtained from the *Seller's organization*, or if this *Seller* defines the *Product\_Root*, then from the supplier Sites.

This Quantity is converted to the unit of the product or product\_group:

Default: 0

**lock\_allocated** - - *a Logical field of model Forecast\_Entry*

If *lock\_allocated* is "true", then the *organization's allocation\_policy* will not adjust this *allocated* Quantity. If *lock\_allocated* is "false", then this *allocated* Quantity will be adjusted according to the *organization's allocation\_policy*. See the *allocated* field and the *Product allocation\_policy* field for further discussion.  
Default: false

**cumulative\_allocated** - - *a Quantity field of model Forecast\_Entry*  
The sum of *allocated* for each *Forecast\_Entry* from the first through this one.

This Quantity is converted to the unit of the product or product\_group:

Properties: Export-Only Field

**specifics\_allocated** - - *a Quantity field of model Forecast\_Entry*  
The sum of the allocated quantities for these *delivery\_dates* for the *specific\_forecasts* (the *Forecasts* for this *Product's specifics\_products*). This will be zero if this is a *GROUP Forecast* or if this *Product* is not a *Generic\_Product* (if it has no *specific\_products*).

Note that *allocated* cannot be set to less than *specifics\_allocated*, and that the *allocated* value will be automatically increased such that it is always at least as much as *specifics\_allocated*, since the *specific\_forecasts* are essentially the subsets of the generic demand modelled by this *Forecast*. Typically, however, *allocated* is more than *specifics\_allocated* due to the uncertainty in the demand for specific products.

This Quantity is converted to the unit of the product:

Properties: Export-Only Field

Models	Forecast_Entry Model
--------	----------------------

members\_accepted - - a Quantity field of model Forecast\_Entry

The sum of the 'accepted' quantities for these 'delivery\_dates' for this Product by the 'member\_forecasts' (the Forecasts of the Sellers that are 'members' of this Seller). This will be zero if this Seller has no 'members'.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.

Properties: Export-Only Field

date\_accepted - - a Date field of model Forecast\_Entry

The Date that the 'accepted' value was last set. Setting 'accepted' will set 'date\_accepted' to 'max(date\_accepted, now)'.

If 'date\_allocated' is later than 'date\_accepted', then an ALLOCATION\_NOT\_ACCEPTED Problem will be created to indicate that a different allocation has been made, but has not yet been accepted or rejected. Rejecting it is as simple as setting 'date\_accepted' to 'now'. Accepting it can be done by 'accept\_as\_allocated'.

Default: the Date when 'accepted' was set (initially infinite past)

accepted - - a Quantity field of model Forecast\_Entry

The Quantity of this 'product' or 'product\_group' for which Promises have been allocated to this Seller for these 'delivery\_dates'. This could also be called "gross ATP" -- the ATP prior to being consumed by actual Promises. The 'available\_to\_promise' is this 'accepted' minus the 'consumed' (the actual Promises that consume from this Forecast\_Entry).

If 'allocated' is less than 'committed', then additional allocation may be obtained from the Seller's 'organization', or if this Seller defines the Product\_Root, then from the supplier Sites.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.

Default: 0

cumulative\_accepted - - a Quantity field of model Forecast\_Entry

The sum of 'accepted' for each Forecast\_Entry from the first through this one.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.

Properties: Export-Only Field

Models	Forecast_Entry Model
--------	----------------------

members\_accepted - - a Quantity field of model Forecast\_Entry

The sum of the 'accepted' quantities for these 'delivery\_dates' for this Product by the 'member\_forecasts' (the Forecasts of the Sellers that are 'members' of this Seller). This will be zero if this Seller has no 'members'.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.

Properties: Export-Only Field

accept\_as\_allocated - - a Void field of model Forecast\_Entry

Accept the allocated values in this Forecast\_Entry and then 'accept\_as\_allocated' in the corresponding Forecast\_Entry of any specific\_forecasts, and similarly 'accept\_as\_allocated' the corresponding Forecast\_Entry(s) of any 'member\_forecasts'. This copies the 'allocated' and 'retain\_from\_allocated' fields to the 'accepted' and 'retain\_from\_accepted' fields in this Forecast\_Entry and every Forecast\_Entry below it in the specific and member trees below it. It also sets 'date\_accepted' in all of those models to 'now'.

Note that the full hierarchy is done because it is in general a bad idea to mix one set of allocations with another -- they may be completely inconsistent. However, you can always set any or all of the fields directly.

Properties: command=True Export-Only Field

consumed - - a Quantity field of model Forecast\_Entry

The total Quantity of this Product for which actual Promises have been made for these 'delivery\_dates', consuming this Forecast\_Entry's 'accepted' allocation.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.

Properties: Export-Only Field

cumulative\_consumed - - a Quantity field of model Forecast\_Entry

The sum of 'consumed' for each Forecast\_Entry from the first through this one.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.

Properties: Export-Only Field

specifics\_consumed - - a Quantity field of model Forecast\_Entry

The sum of the 'consumed' quantities for these 'delivery\_dates' for the 'specific\_forecasts' (the Forecasts for this Product's 'specific\_products'). This will be zero if this is a GROUP Forecast or if this Product is not a Generic\_Product (if it has no 'specific\_products').

Models	Forecast_Entry Model
--------	----------------------

Note that 'consumed' includes the 'specifics\_consumed'. Each actual promise made whose 'consumed\_forecast' is one of the 'specific\_forecasts' will increase 'consumed' not only of that specific Forecast, but also for each of its generic Forecasts (and each of their generic Forecasts, and so on).

This Quantity is converted to the 'unit' of the 'product'.

Properties:    Export-Only Field

**members\_consumed** - - *a Quantity field of model Forecast\_Entry*

The total Quantity of this Product for which actual Promises have been made for these 'delivery\_dates' by the 'members' of this Seller. The sum of the 'consumed' quantities for these 'delivery\_dates' for this Product from the 'member\_forecasts' (the Forecasts of the Sellers that are 'members' of this Seller). This will be zero if this Seller has no 'members'.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.

Properties:    Export-Only Field

**actual\_promises** - - *a List(Item\_Promise) field of model Forecast\_Entry*

The actual (customer) Item\_Promises that have consumed the 'allocated' Quantity of this Forecast\_Entry. The sum of these are added to the combination (not necessarily sum) of the 'specifics\_consumed' and the 'members\_consumed' and to form the 'consumed' of this Forecast\_Entry. (Note that the sum of 'specifics\_consumed' and 'members\_consumed' may double-count the 'actual\_promises' of the 'specific\_forecasts' within the member Seller\_Plans.)

Properties:    Export-Only Field

**available\_to\_promise** - - *a Quantity field of model Forecast\_Entry*

This is 'accepted' - 'consumed'. It is the total Quantity of this 'product' or 'product\_group' that is planned to be built, allocated to this Seller, and unconsumed, such that it is available to the Seller to be promised to customers for these 'delivery\_dates'.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.

The field 'available\_to\_promise' is an alias of 'available' -- provided to match the commonly used term -- but 'available' is more consistent with the family of fields.

Properties:    Export-Only Field

Models	Forecast_Entry Model
--------	----------------------

**available** - - *a Quantity field of model Forecast\_Entry*

This is 'accepted' - 'consumed'. It is the total Quantity of this 'product' or 'product\_group' that is planned to be built, allocated to this Seller, and unconsumed, such that it is available to the Seller to be promised to customers for these 'delivery\_dates'.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.

The field 'available\_to\_promise' is an alias of 'available' -- provided to match the commonly used term -- but 'available' is more consistent with the family of fields.

Properties:    Export-Only Field

**cumulative\_available** - - *a Quantity field of model Forecast\_Entry*

The sum of 'available\_to\_promise' for each Forecast\_Entry from the first through this one.

This is the cumulative Quantity of this 'product' or 'product\_group' over all previous Forecast\_Entries that is planned to be built, allocated to this Seller, and unconsumed, such that it is available to the Seller to be promised to customers for these 'delivery\_dates'.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.

Properties:    Export-Only Field

**members\_available** - - *a Quantity field of model Forecast\_Entry*

The sum of the 'available\_to\_promise' quantities for these 'delivery\_dates' for this Product by the 'member\_forecasts' (the Forecasts of the Sellers that are 'members' of this Seller). This will be zero if this Seller has no 'members'.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.

Properties:    Export-Only Field

**specifics\_available** - - *a Quantity field of model Forecast\_Entry*

The sum of the 'available\_to\_promise' quantities for these 'delivery\_dates' for the 'specific\_forecasts' (the Forecasts for this Product's 'specific\_products'). This will be zero if this is a GROUP Forecast, or if this Product is not a Generic\_Product (if it has no 'specific\_products').

This Quantity is converted to the 'unit' of the 'product'.

Properties:    Export-Only Field

<i>Models</i>	<i>Forecast_Entry Model</i>
---------------	-----------------------------

**allocated\_available** - - *a Quantity field of model Forecast\_Entry*  
 This is simply 'allocated - consumed'. It is the total Quantity of this 'product' or 'product\_group' that would be available to the Seller to be promised to customers for these 'delivery\_dates' if the currently 'allocated' quantities were accepted. In a sense, this is the "what-if" available\_to\_promise.

This Quantity is converted to the unit of the 'product' or 'product\_group'.  
 Properties:   Export-Only Field

**cumulative\_allocated\_available** - - *a Quantity field of model Forecast\_Entry*  
 The sum of 'allocated\_available' for each Forecast\_Entry from the first through this one.

This is the cumulative Quantity of this 'product' or 'product\_group' over all previous Forecast\_Entries would be available to the Seller to be promised to customers for these 'delivery\_dates' if the currently 'allocated' quantities were accepted. In a sense, this is the "what-if" cumulative\_available.

This Quantity is converted to the unit of the 'product' or 'product\_group'.  
 Properties:   Export-Only Field

**planned\_available** - - *a Quantity field of model Forecast\_Entry*  
 This is simply 'planned - consumed'. It is the total Quantity of this 'product' or 'product\_group' that would be available to the Seller to be promised to customers for these 'delivery\_dates' if the currently 'planned' allocations were promised and accepted. In a sense, this is the "what-if" available\_to\_promise.

This Quantity is converted to the unit of the 'product' or 'product\_group'.  
 Properties:   Export-Only Field

**cumulative\_planned\_available** - - *a Quantity field of model Forecast\_Entry*  
 The sum of 'planned\_available' for each Forecast\_Entry from the first through this one.

This is the cumulative Quantity of this 'product' or 'product\_group' over all previous Forecast\_Entries would be available to the Seller to be promised to customers for these 'delivery\_dates' if the currently 'planned' allocations were promised and accepted. In a sense, this is the "what-if" cumulative\_available.

This Quantity is converted to the unit of the 'product' or 'product\_group'.  
 Properties:   Export-Only Field

<i>Models</i>	<i>Forecast_Entry Model</i>
---------------	-----------------------------

**zero\_available\_to\_promise** - - *a Logical field of model Forecast\_Entry*  
 If 'zero\_available\_to\_promise' is "true", then this Forecast\_Entry will never allow 'available\_to\_promise' to be greater than zero. If a situation which would cause 'available\_to\_promise' to be greater than zero occurs, this Forecast\_Entry will automatically release any 'allocated' quantities to its 'organization' in order to force 'available\_to\_promise' to be zero. If 'zero\_available\_to\_promise' is "false", then 'available\_to\_promise' will work normally for this Forecast\_Entry.  
 Default:   false

**allocate\_allocated\_available** - - *a Void field of model Forecast\_Entry*  
 This command allocates any quantity in 'allocated\_available' to the member sellers  
 Properties:   command=True   Export-Only Field

**forecast\_policy** - - *an extension selector of model Forecast\_Entry*  
 This is defined by the 'product's forecast\_policy'. The 'forecast\_policy' determines the additional fields that are forecasted per entry.

Default:   SINGLE\_REQUEST  
 Properties:   Export-Only Field

**allocation\_policy** - - *an extension selector of model Forecast\_Entry*  
 This is defined by the 'product's allocation\_policy'. The 'allocation\_policy' determines the additional fields that are forecasted per entry.

Default:   PER\_COMMITTED  
 Extensions:  
 MEMBER\_RANK.

**forecast\_requests** - - *a List(Item\_Request) field of model Forecast\_Entry*  
 The forecast Item\_Request generated by this Forecast\_Entry's 'committed' value. Note that this will only be populated for the Seller\_Plan of the Seller that owns this Product. If this Product is owned by an 'organization' of this Seller, then this Seller will receive its 'allocated' value from its 'organization' and this list will be empty.  
 Properties:   Export-Only Field

**forecast\_promises** - - *a List(Item\_Promise) field of model Forecast\_Entry*  
 The forecast Item\_Promises corresponding to 'forecast\_requests'. If this Seller owns this Product, then the sum of these Item\_Promises form 'allocated'. This is equivalent to 'forecast\_requests\_for\_each(#.item\_promise)'.  
 Properties:   Export-Only Field

**owner** - - *a Forecast field of model Forecast\_Entry*

Properties: Export-Only Field

5.1.3.2.1.3 level submodels of model Forecast  
5.1.3.2.1.4 ATP\_Entry Model

ATP\_Entry -- a submodel of model Forecast

In each Forecast, there is one ATP\_Entry for each Date\_Range in the Seller\_Plan's list of Date\_Ranges, 'atp\_horizon'. Each ATP\_Entry has the Date\_Range for which it maintains allocations, an 'allocated' value that it can promise, a 'consumed' value that maintains the amount already promised, and an 'atp' value that is still available to promise.

The ATP\_Entry model has fields that references these models :  
Forecast.

These models have a field that is a ATP\_Entry model :  
INDIVIDUAL.

The key field for this model is available\_dates

available\_dates -- a Date\_Range field of model ATP\_Entry  
The dates containing allocations in this ATP\_Entry. This cannot be set -- it is determined by the Seller's 'atp\_horizon' extension. All allocation must be done for the same Date\_Ranges for them to be comparable and aggregable.

This field is not really settable (to be read-only soon).  
Default: None -- this is a key field  
Properties: Export-Only Field

allocated -- a Quantity field of model ATP\_Entry  
The Quantity of this 'product' for which forecast Promises have been allocated to the top-seller for these 'available\_dates'. This could also be called "preliminary gross ATP" -- the ATP prior to being consumed by actual Promises. The ATP is preliminary since the forecast Promises have not yet been accepted. The 'allocated available to promise' is this 'allocated' minus the actual Promises that have been made in a Forecast\_Entry by consuming this allocation.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.  
Default: 0

consumed -- a Quantity field of model ATP\_Entry  
The total Quantity of this Product for which actual Promises have been made for these 'available\_dates', consuming this ATP\_Entry's 'allocated' quantity.

Models	ATP_Entry Model
--------	-----------------

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.  
Default: 0  
Properties: Export-Only Field

*allocated\_available* - - *a Quantity field of model ATP\_Entry*  
This is simply 'allocated - consumed'. It is the total Quantity of this 'product' or 'product\_group' that would be available to the Seller to be promised to customers for these 'available\_dates'.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.  
Properties: Export-Only Field

*cumulative\_allocated\_available* - - *a Quantity field of model ATP\_Entry*  
This Quantity represents the cumulative *allocated\_available* for these 'available\_dates' that would be available to the Seller to be promised to customers for these 'available\_dates'.

This Quantity is converted to the 'unit' of the 'product' or 'product\_group'.  
Properties: Export-Only Field

*owner* - - *a Forecast field of model ATP\_Entry*

Properties: Export-Only Field

Models	Problem Model
--------	---------------

### 5.1.3.3 Problem Model

*Problem* -- *a submodel of model Plan*

The Problem models a violation in a Plan of a Supply\_Chain. Problems include both feasibility problems (things that must be eliminated for the Plan to be valid; such as use of non-existent capacity) and desirability problems (things that would be preferable to eliminate, but not necessary to use the plan; such as late orders). A desirability problem in one company, may be a feasibility problem in another -- thus that division is user definable in many cases.

The model has selectors:  
category.

The Problem model has fields that references these models :  
Plan.

These models have a field that is a Problem model :  
Plan, Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance, Active\_Problem, Item\_Request, Item\_Acceptance, Item\_Promise.

The key field for this model is description

*description* - - *a Computed\_String field of model Problem*  
A textual description of the Problem.

Default: None -- this is a key field  
Properties: Export-Only Field

*dates* - - *a Date\_Range field of model Problem*  
The Date\_Range over which this Problem occurs.  
Properties: Export-Only Field

*feasible* - - *a Logical field of model Problem*  
If "false", then this models a feasibility Problem -- the Plan is not feasible. If "true", then this models an undesirable, but feasible condition.  
Properties: Export-Only Field

*cost* - - *a Money field of model Problem*  
The cost incurred due to this Problem remaining. If 'feasible' is "false", then this 'cost' is infinite.  
Properties: Export-Only Field

last\_change -- a Date field of model Problem  
The Date for which the last change was made for this problem. This field is used by the bookmark facility to determine which problems have been modified since the bookmark was established. It is settable, which can be useful when working with certain Strategies, Problem\_Seis, or Problem-solving Reports.  
Default: the Date that the Problem was created or last modified

category -- an extension selector of model Problem  
The category of Problem. In addition to categorization, it also determines the fields that define what has the Problem and how severe the Problem is. For example, an OVERLOAD Problem will have a Resource\_Plan (what is overloaded) and an 'excess\_load' field which defines how much overload occurs during the dates'.

Note that these same extensions and fields appear in the Problem\_Seis submodel of Strategy for specifying which Problems the Strategy should be working on.  
Properties: Export-Only Field

Extensions:  
REQUEST\_NOT\_PLANNED, REQUEST\_PLANNED\_LATE,  
REQUEST\_PLANNED\_EARLY, REQUEST\_PLANNED\_SHORT,  
REQUEST\_PLANNED\_EXCESS, PROMISE\_NOT\_PLANNED,  
PROMISE\_PLANNED\_LATE, PROMISE\_PLANNED\_EARLY,  
PROMISE\_PLANNED\_SHORT, PROMISE\_PLANNED\_EXCESS,  
ACCEPTANCE\_NOT\_PLANNED, ACCEPTANCE\_PLANNED\_LATE,  
ACCEPTANCE\_PLANNED\_EARLY, ACCEPTANCE\_PLANNED\_SHORT,  
ACCEPTANCE\_PLANNED\_EXCESS, PLANNED\_BEFORE\_CURRENT,  
UNRELEASED, NEEDS\_RELEASE, INCONSISTENT\_OPPLAN,  
REQUEST\_PROMISED\_LATE, REQUEST\_PROMISED\_EARLY,  
REQUEST\_PROMISED\_SHORT, REQUEST\_PROMISED\_EXCESS,  
ITEM\_PROMISE\_OVERPRICED, DELIVERY\_PROMISE\_OVERPRICED,  
PROMISE\_NOT\_OFFERED, PROMISE\_NOT\_ACCEPTED,  
ACCEPTANCE\_INCONSISTENT, REQUEST\_QUEUED,  
DELIVERY\_REQUEST\_NOT\_COORDINATED,  
DELIVERY\_PROMISE\_NOT\_COORDINATED,  
DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED, NEGATIVE\_ATP,  
NEGATIVE\_PLANNED\_ATP, OVER\_COMMITTED, OVER\_CONSUMED,  
UNALLOCATED\_FORECAST, SUPPLY\_PLANNED\_LATE,  
SUPPLY\_PLANNED\_EARLY, SUPPLY\_PLANNED\_SHORT,  
SUPPLY\_PLANNED\_EXCESS, SUPPLY\_PROMISED\_LATE,  
SUPPLY\_PROMISED\_EARLY, SUPPLY\_PROMISED\_SHORT,  
SUPPLY\_PROMISED\_EXCESS, UNIDENTIFIED\_OP\_STATE, UNCONSOLIDATED,  
UNCOORDINATED, CONSOLIDATION\_OVERSIZE,

CONSOLIDATION\_UNDERSIZE, OVERLOAD, OVERSIZE,  
BUCKET\_OVERSIZE, UNDERLOAD, NEGATIVE\_ON\_HAND,  
OVER\_FLOW\_LIMIT, NEGATIVE\_ON\_HAND\_AT\_END,  
LOT\_OVER\_CONSUMED, LOT\_NOT\_CONSUMED,  
LOT\_NOT\_PRODUCED, LOT\_OVER\_PRODUCED, LOW\_ON\_HAND,  
EXCESS\_ON\_HAND, EXCESS\_ON\_HAND\_AT\_END,

in\_category (Problem\_Category) -- a Logical field of model Problem  
Returns "true" if this Problem is in the specified Problem\_Category.  
Properties: Export-Only Field

interaction -- a Number field of model Problem  
Heuristic estimate of the interaction of this Problem with other Problems and potential Problems. The higher this number, the more difficult it will likely be to resolve this Problem without creating other Problems.  
Properties: Export-Only Field

resolvable -- a Logical field of model Problem  
If "false", then this Problem has been determined to be unresolvable by the automated solvers -- human intervention will be needed.  
Properties: Export-Only Field

resolve, resolve (Active\_Strategy) -- a Void field of model Problem  
Attempts to eliminate this Problem under the direction of the currently 'running' Active\_Strategy of the 'owner' Plan. If there is no Active\_Strategy 'running', then it uses the 'auto\_run' Active\_Strategy. The resolve(Active\_Strategy) function resolves a problem under the direction of the specified active strategy (whether it is 'running' or not).  
Properties: command=True Export-Only Field

owner -- a Plan field of model Problem

Properties: Export-Only Field



Models	Active_Strategy Model
--------	-----------------------

### 5.1.3.4 Active\_Strategy Model

Active\_Strategy -- *a submodel of model Plan*

The Strategy's that remain active and up-to-date on every Plan change.

The Active\_Strategy models an approach to resolving the remaining Problems in a Plan. It consists of a specification of what Problems to attempt to resolve, what modifications can be made in those attempts, and what criteria to use for judging the "goodness" of the Plan.

The model has selectors:  
termination, execution, problem\_selection.

The Active\_Strategy model has these submodels :  
Active\_Problem, Active\_Goal.

The Active\_Strategy model has fields that references these models :  
Strategy, Active\_Strategy, Active\_Problem, Active\_Goal, Plan.

These models have a field that is a Active\_Strategy model :  
Plan, Active\_Strategy, Active\_Problem, Active\_Goal,  
BEFORE\_AND\_AFTER, SEQUENTIAL\_ALTERNATES,  
SEQUENTIAL\_ALTERNATES\_KEEP\_BEST.

The key field for this model is strategy  
This model may be extended with user-defined fields.

strategy -- *a Strategy field of model Active\_Strategy*  
The Strategy...  
Default: None -- this is a key field

super -- *a Active\_Strategy field of model Active\_Strategy*  
The strategy this is a sub of  
Properties: java\_method=parent Export-Only Field  
remark -- *a String field of model Active\_Strategy*  
Remarks about the performance of the 'strategy' on the 'owner' Plan.  
Default: none

Models	Active_Strategy Model
--------	-----------------------

date\_activated -- *a Date field of model Active\_Strategy*

The Date at which this Active\_Strategy was activated for this Plan. Note that some Problem\_Sets of the Strategy may specify that only Problems changed after the Strategy was activated should be addressed. (See the 'last\_change\_after\_activated' field of Problem\_Set.) In that case, only Problems changed after this 'date\_activated' would be addressed. This allows the user to make some changes and then work only the effects of those changes.

Properties: Export-Only Field

reset\_date\_activated -- *a Date field of model Active\_Strategy*

Resets the activation date of the strategy to be the current date. This function incurs a 2 second delay. The delay is necessary so that there won't be any problems modified at the same time as the activation date is reset.

Properties: command=True Export-Only Field

run -- *a Void field of model Active\_Strategy*

Run this Active\_Strategy, starting with the first sub\_strategy. Try to resolve the specified Problems, according to the specified conditions.

Properties: command=True Export-Only Field

stop -- *a Void field of model Active\_Strategy*

Stop this Active\_Strategy. Only has an effect if it is currently 'running'.  
Properties: command=True; exec\_priority=IMMEDIATE Export-Only Field

continue -- *a Void field of model Active\_Strategy*

Continue this Active\_Strategy from where it was last 'stop'd.  
Properties: command=True; java\_method=continue\_running Export-Only Field

running -- *a Logical field of model Active\_Strategy*

If "true", this 'strategy' is currently running on the 'owner' Plan. Setting this to "true" is equivalent to calling 'run'; setting it to "false" returns it to its initial state.

Default: false  
Properties: Export-Only Field

stopped -- *a Logical field of model Active\_Strategy*

If "true", this 'strategy' is 'running' but has been stopped (via 'stop'). Setting this to "true" is equivalent to calling 'stop'; setting it to "false" is equivalent to calling 'continue'. If this 'strategy' is not 'running', then this is necessarily "false", and setting it has no effect.

Default: false  
Properties: Export-Only Field

**run\_time** - - *a Time field of model Active\_Strategy*  
The total run Time that has elapsed. This is compared to 'strategy\_max\_run\_time'.  
Properties: Export-Only Field

**stable\_time** - - *a Time field of model Active\_Strategy*  
The total Time that has elapsed since the last improvement was made to the Plan. This is compared to 'strategy\_max\_stable\_time'.  
Properties: Export-Only Field

**target\_achieved** - - *a Logical field of model Active\_Strategy*  
Returns "true" if the target as specified by the 'strategy\_goal' and related fields has been achieved. Note that the ultimate goal may still not be achieved, though the target has been.  
Properties: Export-Only Field

**interaction** - - *a Number field of model Active\_Strategy*  
Returns the sum of interaction values from all active problems  
Properties: Export-Only Field

**annealing\_goodness** - - *a Number field of model Active\_Strategy*  
Returns a number representing the comparative goodness of the plan. Initially the annealing\_goodness will return 1. Smaller goodness indicates a better plan, judged by its goals.  
Properties: Export-Only Field

**resolve\_count** - - *a Number field of model Active\_Strategy*  
The number of resolves that have been called by the strategy so far.  
Properties: Export-Only Field

**permanent** - - *a Logical field of model Active\_Strategy*  
If "true", then this 'strategy' remains active on the Plan even when not running. The Problem Lists and performance information is maintained for every change to the Plan. If "false", then this Active\_Strategy will vanish once it completes running. Thus, the only way to see a non-permanent Active\_Strategy is to run a Strategy on this Plan and then observe it while it is still running.  
Default: true

**auto\_run** - - *a Logical field of model Active\_Strategy*  
If "true", the Strategy is run after each change to the 'owner' Plan. If "false", this Active\_Strategy is kept up-to-date, but not run on each change.

There can be only one Active\_Strategy for any Plan that has 'auto\_run' set to "true". Setting this to "true" will set all other Active\_Strategy's of this Plan to "false".  
Default: false

**background\_run** - - *a Logical field of model Active\_Strategy*  
If "true", the Strategy is 'run' as a background activity whenever there has been no planning activity. In this way you can have Strategy's that churn all day trying to find small improvements to the Plan.

Note that when planning activity resumes, the background Strategy will need to stop running first. That may cause a noticeable delay on that first access.

There should be at most one Active\_Strategy for any Plan that has 'background\_run' set to "true". Setting this to "true" will set all other Active\_Strategy's of this Plan to "false".  
Default: false

**active\_problems** - - *a list of Active\_Problem submodels of model Active\_Strategy*  
The Problems in the 'owner' Plan that this Active\_Strategy is working to resolve and the 'Focus' that it is giving to each, as defined by one of the Problem\_Sets in the 'strategy'. Note that if a Problem is selected by more than one Problem\_Set, the highest 'Focus' value is used.

**problems**, **problems (Date\_Range)**, **problems (Problem\_Category)**, **problems (Date\_Range, Problem\_Category)** - - *a List[Problem] field of model Active\_Strategy*  
The Problems detected with this Plan. If passed a Date\_Range, only the Problems whose 'dates' overlap are returned. If passed a Problem\_Category, only the Problems with that 'category' are returned.

Note that you can pass in one of the special Problem\_Category's to get Problems in larger groups. For example, the OPERATION Problem\_Category will give all Problems in Operation-related Problem categories. Similarly for RESOURCE, BUFFER, and even ALL.  
Properties: Export-Only Field

**problem\_categories** - - *a List[Problem\_Category] field of model Active\_Strategy*  
For each different 'category' of Problem in 'problems', a Problem\_Category is added to this list. It gives you the name of the 'category' (in various forms) plus a list of just the Problems of that 'category'. These sublists are often much easier to deal with than the full 'problems' list.

Properties:    Export-Only Field

**active\_goals** - - *a list of Active\_Goal submodels of model Active\_Strategy*  
The goals of this Active\_Strategy and focus for each as defined by the 'strategy'. In addition, the current value(s) of this Plan for each goal.

**termination** - - *an extension selector of model Active\_Strategy*

Defines the Active\_Strategy's progress towards termination, as defined by the 'strategy'. For example, it may be specify to stop after N seconds, to stop after N seconds of no Plan improvements, to stop when the next improved Plan is found, to stop when a certain Cost goal is achieved, etcetera.

Properties:    Export-Only Field

Extensions:  
**NO\_PROBLEMS, TARGET\_ACHIEVED, RESOLVE\_COUNT\_EXCEEDED, MANUAL.**

**execution** - - *an extension selector of model Active\_Strategy*  
execution specifies when and where the sub strategies are called

Properties:    Export-Only Field

Extensions:  
**SEQUENCE, RUN\_ONCE, SEQUENCE, RUN\_MULTIPLE, BEFORE\_AND\_AFTER, SEQUENTIAL\_ALTERNATES, SEQUENTIAL\_ALTERNATES\_KEEP\_BEST, PROPORTIONAL\_RESOLVES, ORDERED\_RESOLVES.**

**problem\_selection** - - *an extension selector of model Active\_Strategy*

The problem\_selection extension specifies how this strategy will select problems to solve.

Properties:    Export-Only Field

Extensions:  
**PROPORTIONAL\_INTERACTION, EARLIEST\_PROBLEM\_START, SORT\_BY\_EXPRESSION.**

**owner** - - *a Plan field of model Active\_Strategy*

Properties:    Export-Only Field

5.1.3.4.1 Active\_Problem Model

**Active\_Problem** - - *a submodel of model Active\_Strategy*

The Problems in the 'owner' Plan that this Active\_Strategy is working to resolve and the 'focus' that it is giving to each, as defined by one of the Problem\_Sets in the 'strategy'. Note that if a Problem is selected by more than one Problem\_Set, the highest 'focus' value is used.

The Active\_Problem model has fields that references these models :  
**Problem, Active\_Strategy.**

These models have a field that is a Active\_Problem model :  
**Active\_Strategy.**

The key field for this model is problem

**problem** - - *a Problem field of model Active\_Problem*

A Problem to be addressed by this Active\_Strategy with 'focus'.

Default:    None - - this is a key field

Properties:    Export-Only Field

**focus** - - *a Percentage field of model Active\_Problem*

A Percentage that indicates how much focus will be given to the 'problem' by this Active\_Strategy. The greater the Percentage, the more attention that the 'problem' will get.

This is computed from the Strategy's 'problem\_sets' and 'feasible\_focus'.

Properties:    Export-Only Field

**unresolvable** - - *a Logical field of model Active\_Problem*

Set to true when the strategy detects that this problem is not resolvable using the current change categories. Site edits, and in some cases changes to the plan, may cause the problem to become resolvable again (but in many cases this field will not be updated for some time).

Default:    false

Properties:    Export-Only Field

**interaction** - - *a Number field of model Active\_Problem*

A number indicating the impact on annealing goodness. It's value depends on problem size and Active\_Strategy's goals.

Properties:    Export-Only Field

**must\_resolve** - - *a Logical field of model Active\_Problem*

Indicates whether this is a must\_resolve problem. This field is true if 'problem' is a member of at least one Problem\_Set whose must\_resolve is true.

Default:    false

Properties:    Export-Only Field

**owner** - - *a Active\_Strategy field of model Active\_Problem*

Properties:    Export-Only Field

5.1.3.4.2    Active\_Goal    Model

**Active\_Goal** -- *a submodel of model Active\_Strategy*

The goals of this Active\_Strategy and focus for each as defined by the 'strategy'. In addition, the current value(s) of this Plan for each goal.

The model has selectors:

**goal.**

The Active\_Goal model has fields that references these models :  
**Strategy\_Goal, Active\_Strategy.**

These models have a field that is a Active\_Goal model :  
**Active\_Strategy.**

The key field for this model is **strategy\_goal**

**strategy\_goal** - - *a Strategy\_Goal field of model Active\_Goal*

The goal defined in the Strategy for which this models the current value for this Plan.

Default:    None -- this is a key field

Properties:    Export-Only Field

**goal** - - *an extension selector of model Active\_Goal*

This is defined by 'strategy\_goal.goal'. This extension will add fields that contain the absolute values for the current value for this Plan, the values before adjustment to a comparable numeric value.

Properties:    Export-Only Field

Extensions:

**FEASIBILITY, MINIMIZE\_PROBLEM\_COUNT, MINIMIZE\_PROBLEMS, MINIMIZE\_LATENCY, WEIGHTED\_LATENCY, WEIGHTED\_SHORTNESS, MINIMIZE\_SHORTNESS, MINIMIZE\_COST, MAXIMIZE\_PROFIT, MAXIMIZE\_REVENUE.**

**adjusted\_value** - - *a Number field of model Active\_Goal*

The adjusted numeric value of this 'goal' for the Plan, comparable to the 'adjusted\_target' for this goal and the 'adjusted\_value's of other goals.

Properties:    Export-Only Field

Models	Active_Goal Model
--------	-------------------

**adjusted\_target** - - *a Number field of model Active\_Goal*  
 The target value of this 'goal', after adjustments. There will often be an extension-specific target field that is before adjustments.

Achieving values higher than the target may receive much less focus ('focus\_beyond\_target') than achieving the target ('focus\_to\_target').  
 Properties:   Export-Only Field

**focus\_value** - - *a Number field of model Active\_Goal*  
 This is the adjusted numeric value weighted by the 'target\_focus' and 'over\_target\_focus' as it will be combined with the other goals. The sum of the 'focus\_value's for all the goals is the overall "global goodness" value.

If 'adjusted\_value' is less than 'adjusted\_target', this value is simply 'focus\_to\_target \* adjusted\_value'. If 'adjusted\_value' is over 'adjusted\_target', then this value is 'focus\_beyond\_target \* (adjusted\_value - adjusted\_target) + focus\_to\_target \* adjusted\_target'.  
 Properties:   Export-Only Field

**owner** - - *a Active\_Strategy field of model Active\_Goal*  
 Properties:   Export-Only Field

Models	Problem_Category Model
--------	------------------------

## 5.2 Problem\_Category Model

**Problem\_Category** -- *an independent (top-level) model*

A Problem\_Category models a category of Problems. It is not a Problem, nor does it contain Problems. Rather, it tells you about a kind of Problem. It is a lot like the Model\_Type model, which models a type of model.

The Problem\_Category model provides the 'name', the 'short\_name', the 'long\_name', and a 'description' of the nature of Problems of that category. Further, it provides a list of the Problem\_Field's that are specific to that category.

Each planning model that provides a List[Problem] also provides a List[Problem\_Category]. Those Problem\_Category's can be passed to the field to get just the Problems in that Problem\_Category. These sublists are often much easier to deal with than the full list of Problems. In particular, coupled with the list of the fields in this Problem\_Category, formulating informative Problem Reports is much easier.

There are several special Problem\_Category's that are not really Problem 'category' extensions, but rather a set of Problem 'category's. For example, the Problem\_Category\_OPERATION includes all Problem 'category's that relate to Operations. Similarly for RESOURCE, BUFFER, and ALL. ALL includes all Problem 'category's.

The key field for this model is name  
 This model may be extended with user-defined fields.

**name** - - *a Symbol field of model Problem\_Category*  
 This is the name used as the 'category' extension selector value in the Problem model.  
 Default:   None -- this is a key field  
 Properties:   Export-Only Field

**short\_name** - - *a Symbol field of model Problem\_Category*  
 The short name for the Problem 'category' extension.  
 Properties:   Export-Only Field

**full\_name** - - *a Symbol field of model Problem\_Category*  
 The full name for the Problem 'category' extension.  
 Properties:   Export-Only Field

Models	Problem_Category Model
--------	------------------------

description - - *a String field of model Problem\_Category*  
 A description of the Problem 'category' extension.  
 Properties:   Export-Only Field

remark - - *a String field of model Problem\_Category*  
 Additional user-provided discussion about this Problem 'category' extension.  
 Default:   none

super\_categories - - *a List(Problem\_Category) field of model Problem\_Category*  
 The direct super\_categories of this Problem\_Category.  
 Properties:   Export-Only Field

sub\_categories - - *a List(Problem\_Category) field of model Problem\_Category*  
 The direct sub\_categories of this Problem\_Category.  
 Properties:   Export-Only Field

leaf\_categories - - *a List(Problem\_Category) field of model Problem\_Category*  
 The leaf sub\_categories of this Problem\_Category. A leaf Problem\_Category is one that contains no other Problem\_Category's; one that identifies only a single 'category' extension of Problem.  
 Properties:   Export-Only Field

fields - - *a List(Symbol) field of model Problem\_Category*  
 The Fields provided by this Problem 'category' extension.  
 Properties:   Export-Only Field

Models	Horizon Model
--------	---------------

### 5.3 Horizon Model

Horizon -- *an independent (top-level) model*

Horizon models how to break up time into buckets. For example, forecasting is often done in monthly buckets, but is not uncommonly done in quarters, weeks, and other variations. Further, there will often be smaller buckets in the near-term followed by larger buckets. For example, weeks for the first 3 months, months for the next 6 months, and then quarters for the rest of the horizon.

Most often the same bucketing is desired in most places. Inconsistent buckets can prevent meaningful comparisons and analyses. However, there is a very real need to be able to choose different bucketing in different areas. Thus, rather than have to define and maintain the same bucket specification in each of hundreds of models, the Horizon model allows you to define named specifications that can then be reused by name in all of those models that should use a common bucket specification.

The model has selectors:  
 bucket\_spec.

These models have a field that is a Horizon model :  
 Seller, Horizon\_Bucket\_Start, HORIZON, SHARED\_USE, BUCKETED\_NESTED\_SORT.

The key field for this model is name  
 This model may be extended with user-defined fields.

name - - *a Symbol field of model Horizon*  
 The name of this Horizon.  
 Default:   None -- this is a key field

description - - *a String field of model Horizon*  
 A description of this Horizon.  
 Default:   none

bucket\_spec - - *an extension selector of model Horizon*

The 'bucket\_spec' extension and related fields define how to compute the time buckets that the planning horizon or Date\_Range should be broken into. It always snaps the ending date to the end of the last bucket, even if the plan horizon ends in the middle of the bucket. Because the last bucket will always be consistent in size with the other buckets, the user will be able to see more easily how changes to the plan horizon affect the buckets. This avoids potential errors which may arise from edits to plan.horizon which move the horizon.end less than one bucket size.

Default: ONE

Extensions:

ONE, MONTHS, WEEKS, DAYS, DATES,

buckets (Date\_Range) - - *a list of Date\_Range fields of model Horizon*

This will return list of buckets defined by currently selected 'bucket\_spec' extension.

If no argument is passed, this returns list of buckets within planning horizon(plan.start to plan.end). Start date for the first bucket will be same as the plan.start and the end date for the last bucket in the list will be same as plan.end. !!!UNIMPLEMENTED!!!

If Date\_Range argument is given, this will return list of buckets within that Date\_Range. First bucket's start will be same as the start date of the Date\_Range passed in as argument. Similarly, the end date of last bucket will be same as the end date of the Date\_Range passed in as argument.  
Properties: Export-Only Field

5.3.1 bucket\_spec submodels of model Horizon  
5.3.2 Horizon\_Bucket\_Start Model

Horizon\_Bucket\_Start -- *a submodel of model Horizon*

List of bucket\_starts specifying bucket boundaries.

The Horizon\_Bucket\_Start model has fields that references these models :  
Horizon.

These models have a field that is a Horizon\_Bucket\_Start model :  
DATES.

The key field for this model is start

start - - *a Date field of model Horizon\_Bucket\_Start*  
start date of the bucket.  
Default: None -- this is a key field

owner - - *a Horizon field of model Horizon\_Bucket\_Start*

Properties: Export-Only Field

Models	Strategy Model
--------	----------------

### 5.4 Strategy Model

Strategy -- *an independent (top-level) model*

An enumeration value used by the Strategy\_Change 'category' field. See that field for details.

The Strategy allows a user to drive the automated planning algorithms. There are five basic things that can be specified by a Strategy: the problems to focus on, the changes to the plan to focus on, the goal trying to be achieved (what is a good plan, what is better), the termination condition (when is this Strategy "done"), and sub-strategies which allows a planning flow to be assembled.

The problems to focus on are specified by the Problem\_Set submodels. Each Problem\_Set defines a category of Problems, possibly some filtering expressions or parameters, and a focus level (which is relative to any other Problem\_Sets). For example, you can specify that the Strategy focuses heavily on BUFFER Problems, lightly on RESOURCE Problems, and not at all on PROMISE\_PLAN Problems. Or in more detail, heavily on NEGATIVE\_ON\_HAND Buffer Problems and moderately on LOW\_ON\_HAND Buffer Problems.

The plan changes to focus on making are specified by the Strategy\_Change submodels. Strategy\_Change submodels define the relative 'focus' that should be given to each category of problem resolutions. For example, to avoid making anything later one can specify that MOVE\_OUT is disallowed.

The goals of this Strategy -- the computation of the quality of a plan -- are specified by the Strategy\_Goal submodels. Each strategy goal defines both a goal (e.g. MAXIMIZE\_REVENUE) and the relative weight of this goal vs. the other goals of the strategy.

When this Strategy is "done" is specified by the 'termination' extension and related fields. For example, one might specify that the strategy is done when a feasible plan is found in which lateness is below a given bound.

Finally, the Sub\_Strategy submodels specify Strategies that are run as part of this Strategy.

The model has selectors: termination, execution, problem\_selection,

The Strategy model has these submodels :

Models	Strategy Model
--------	----------------

Problem\_Set, Strategy\_Change, Strategy\_Lock, Strategy\_Goal.

The Strategy model has fields that references these models :

Problem\_Set, Strategy\_Change, Strategy\_Lock, Strategy\_Goal.

These models have a field that is a Strategy model :

Active\_Strategy, Problem\_Set, Strategy\_Change, Strategy\_Lock, Strategy\_Goal, Ordered\_Sub\_Strategy, BEFORE\_AND\_AFTER, SEQUENTIAL\_ALTERNATES, SEQUENTIAL\_ALTERNATES, KEEP\_BEST, Ranked\_Sub\_Strategy.

The key field for this model is name

name -- *a Symbol field of model Strategy*

The name of this Strategy:

Default: None -- this is a key field

description -- *a String field of model Strategy*

A description of the goals, approach, usage, and usefulness of this Strategy.

Default: no empty String

deterministic\_resolvers -- *a Logical field of model Strategy*

Default FALSE. When TRUE causes the strategy to always set heai=0. This should cause most resolvers to behave deterministically.

Default: false

deterministic\_problem\_selection -- *a Logical field of model Strategy*

Default FALSE. When TRUE causes the strategy to always select the problem for which the product of the interaction and focus value is the largest.

Default: false

problem\_sets -- *a list of Problem\_Set submodels of model Strategy*

Each Problem\_Set defines a set of Problems to be addressed by this Strategy. The Problems are specified by category and tolerances within a certain horizon. The relative 'focus' that the Strategy should place on each Problem is also specified.

changes -- *a list of Strategy\_Change submodels of model Strategy*

Defines the relative 'focus' that should be given to each Change\_Category that can be performed to resolve Problems. For example, to avoid making anything later, you can specify that MOVE\_OUT is disallowed (0% focus).



Change\_Category's that do not appear in this List get the 'default\_change\_focus'.

locks - - *a list of Strategy\_Lock submodels of model Strategy*

A list of specifications that describe what objects are to remain locked/unchanged for the duration of the strategy execution. For example, users can lock Operation\_Plans using Strategy\_Lock. The effects of multiple Strategy\_Locks are additive; should any Strategy\_Lock consider an Operation\_Plan 'locked', then it will be treated as such.

default\_change\_focus - - *a Percentage field of model Strategy*

The 'focus' for Change\_Category's not explicitly defined in 'changes'. By setting this to 0%, no change will be tried unless it is mentioned in 'changes'. By setting this greater than 0%, all changes will be tried unless explicitly given 'focus' of 0% in 'changes'. Default: 100%

goals - - *a list of Strategy\_Coal submodels of model Strategy*

The goals of this strategy. This defines global goodness.

termination - - *an extension selector of model Strategy*

Defines how the Strategy decides to terminate a run. For example, it may be specify to stop after N seconds, to stop after N seconds of no Plan improvements, to stop when the next improved Plan is found, to stop when a certain Cost goal is achieved, etcetera.

Note that by default, the Strategy will run forever!

Default: NO\_PROBLEMS

Extensions:

NO\_PROBLEMS, TARGET\_ACHIEVED, RESOLVE\_COUNT\_EXCEEDED, MANUAL.

execution - - *an extension selector of model Strategy*

The execution extension allows users to specify Sub\_Strategy's that will be performed as part of this Strategy. It will also define how these Sub\_Strategy's will be executed.

For example, SEQUENCE\_RUN\_ONCE execution extension specifics

Sub\_Strategy's that will run to termination in a particular sequence exactly once.

PROPORTIONAL\_RESOLVE execution extension will repeatedly select one of the Sub\_Strategy probabilistically and resolve one of it's Active\_Problem.

A Strategy "run" continues until one of it's termination conditions are met. A "run" consists of a series of "searches". A "search" is a series of Problem "resolves" that result in a new plan. If a search results in a better Plan, where better is defined by the Strategy 'goals', then the search is termed a "success".

Default: SEQUENCE\_RUN\_ONCE

Extensions:

SEQUENCE\_RUN\_ONCE, SEQUENCE\_RUN\_MULTIPLE, BEFORE\_AND\_AFTER, SEQUENTIAL\_ALTERNATES, SEQUENTIAL\_ALTERNATES\_KEEP\_BEST, PROPORTIONAL\_RESOLVES, ORDERED\_RESOLVES,

problem\_selection - - *an extension selector of model Strategy*

The problem\_selection extension specifies how this strategy will select problems to solve.

Default: PROPORTIONAL\_INTERACTION

Extensions:

PROPORTIONAL\_INTERACTION, EARLIEST\_PROBLEM\_START, SORT\_BY\_EXPRESSION.

max\_stable\_time - - *a Time field of model Strategy*

This Strategy will terminate if it does not find an improvement to the plan within

'max\_stable\_time'.

Default: infinite

max\_stable\_resolve\_count - - *a Number field of model Strategy*

This Strategy will advance to the next annealing run or terminates if it could not find an improvement to the plan within 'max\_stable\_resolve\_count' problem resolutions.

Default: infinite

Properties: obsolete=True

max\_run\_time - - *a Time field of model Strategy*

This Strategy will terminate if it runs for longer than 'max\_run\_time'.

Default: infinite

max\_resolve\_count - - *a Number field of model Strategy*

This Strategy will terminate if it attempts more than 'max\_resolve\_count' resolves.

Default: infinite

max\_heat - - *a Quantity field of model Strategy*

Strategy will start every annealing run with this value of maximum heat.

Default: 100

min\_heat - - *a Quantity field of model Strategy*

Lowest annealing heat.

Default: 0.0

Models	Strategy Model
--------	----------------

```

run (Plan) -- a Void field of model Strategy
Run the Strategy. Try to resolve the specified Problems, according to the specified
conditions.
Properties:    command=True Export-Only Field

do_before -- a Expression field of model Strategy
An Expression that is passed an Active_Strategy as 'w' and is executed before the
Strategy run.
Default: NONE

do_after -- a Expression field of model Strategy
An Expression that is passed an Active_Strategy as 'w' and is executed after the Strat-
egy run.
Default: NONE

do_before_resolve -- a Expression field of model Strategy
An Expression that is passed an Active_Problem as 'w' and is executed before the
problem resolution.
Default: NONE

```

Models	Problem_Set Model
--------	-------------------

### 5.4.1 Problem\_Set Model

#### Problem\_Set -- a submodel of model Strategy

Each Problem\_Set defines a set of Problems to be addressed by this Strategy. The Problems are specified by category and tolerances within a certain horizon. The relative focus that the Strategy should place on each Problem is also specified.

The model has selectors:  
category.

The Problem\_Set model has fields that references these models :  
Strategy.

These models have a field that is a Problem\_Set model :  
Strategy.

The key field for this model is category  
This model may be extended with user-defined fields.

```

fence -- a Horizon_Date field of model Problem_Set
This Problem_Set will only identify a Problem with 'dates.start' prior to this 'fence'.
This 'fence' is computed as a Horizon_Date relative to 'plan.current'.
Default: oo_future

```

```

min_time -- a Time field of model Problem_Set
This Problem_Set will only identify a Problem with 'dates.time' >= this 'min_time'.
The 'dates' in the problem do not necessarily reflect the the duration over which prob-
lem exists. In some cases, the dates associated with the problem are the start and end
dates of the flow plan which has the problem. The min_time field cannot be used to
filter problems based on the time between the problem occurring and the end of the
planning horizon -- unless, of course, the source of the problem's dates happens to
accidentally overlap that region. The meaning of min_time varies with problem cate-
gory.
Default: 0

```

```

last_change_after_activated -- a Logical field of model Problem_Set
If "true", then this Problem_Set will only identify Problems with 'last_change' after
'date_activated' of the Active_Strategy.

```

This allows the user to make some changes and then have a Strategy work on only the effects of those changes. Given a Strategy that has Problem\_Sets with last\_change\_after\_activated set "true", you can activate it on a Plan, make changes to that Plan, and then run that Active\_Strategy. Only the Problems created or changed due to the changes made to that Plan since you activated the Strategy will be addressed.

Default: false

category -- an extension selector of model Problem\_Set

A Problem 'category' that this Strategy will address. This extension adds similar fields as the Problem 'category' adds. Here they specify tolerances on what is allowable.

Problems within the tolerance are not addressed by this Strategy.

There are also additional category extensions that are generalizations of the Problem 'category' extensions. For example, RESOURCE will match any Resource\_Plan related Problem, including OVERLOAD, UNDERLOAD, OVERTIME, and the rest. Default: None -- this is a key field

Extensions:

REQUEST\_NOT\_PLANNED, REQUEST\_PLANNED\_LATE, REQUEST\_PLANNED\_EARLY, REQUEST\_PLANNED\_SHORT, REQUEST\_PLANNED\_EXCESS, PROMISE\_NOT\_PLANNED, PROMISE\_PLANNED\_LATE, PROMISE\_PLANNED\_SHORT, PROMISE\_PLANNED\_EARLY, PROMISE\_PLANNED\_EXCESS, ACCEPTANCE\_NOT\_PLANNED, ACCEPTANCE\_PLANNED\_LATE, ACCEPTANCE\_PLANNED\_EARLY, ACCEPTANCE\_PLANNED\_SHORT, ACCEPTANCE\_PLANNED\_EXCESS, PLANNED\_BEFORE\_CURRENT, UNRELEASED, NEEDS\_RELEASE, INCONSISTENT\_OPLAN, OPERATION, REQUEST\_PROMISED\_LATE, REQUEST\_PROMISED\_EARLY, REQUEST\_PROMISED\_SHORT, REQUEST\_PROMISED\_EXCESS, PROMISE\_NOT\_OFFERED, PROMISE\_NOT\_ACCEPTED, ACCEPTANCE\_INCONSISTENT, REQUEST\_QUEUED, REQUEST, REQUEST\_PLAN, PROMISE, PROMISE\_PLAN, REQUEST\_PROMISE, DELIVERY\_REQUEST\_NOT\_COORDINATED, DELIVERY\_PROMISE\_NOT\_COORDINATED, DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED, SUPPLY\_PLANNED\_LATE, SUPPLY\_PLANNED\_EARLY, SUPPLY\_PLANNED\_SHORT, SUPPLY\_PLANNED\_EXCESS, SUPPLY\_PROMISED\_LATE, SUPPLY\_PROMISED\_EARLY, SUPPLY\_PROMISED\_SHORT, SUPPLY\_PROMISED\_EXCESS, SUPPLY\_PLAN, SUPPLY\_PROMISE, UNIDENTIFIED\_OP\_STATE, UNCONSOLIDATED, UNCOORDINATED, CONSOLIDATION\_OVERSIZE, CONSOLIDATION\_UNDERSIZE, OVERLOAD, OVERSIZE,

BUCKET\_OVERSIZE, UNDERLOAD, RESOURCE, NEGATIVE\_ON\_HAND, OVER\_FLOW\_LIMIT, NEGATIVE\_ON\_HAND\_AT\_END, LOT\_OVER\_CONSUMED, LOT\_NOT\_CONSUMED, LOT\_NOT\_PRODUCED, LOT\_OVER\_PRODUCED, LOW\_ON\_HAND, EXCESS\_ON\_HAND, EXCESS\_ON\_HAND\_AT\_END, BUFFER.

min\_cost -- a Money field of model Problem\_Set

This Strategy only addresses Problems with 'cost' >= this 'min\_cost'.

Default: 0

must\_resolve -- a Logical field of model Problem\_Set

Specifies whether problems of this problem set must be resolved. Problem sets for which this field is false are desirable to resolve but not necessary to resolve. SDP behavior is greatly altered by this field.

Default: false

focus -- a Percentage field of model Problem\_Set

A Percentage that indicates how much focus should be given to Problems identified by this Problem\_Set, relative to the other Problem\_Sets in the 'owner' Strategy. The greater the Percentage, the more attention that those Problems will get. If a single Problem is identified by more than one Problem\_Set, the higher focus is used.

Default: 100%

feasible\_focus -- a Percentage field of model Problem\_Set

Weights the focus of 'feasible' Problems relative to Problems that are not 'feasible'. If feasible, the Problem will receive focus equal to 'focus' multiplied by this value. If "0%", then 'feasible' Problems are not addressed at all. If "100%", then 'feasible' Problems are addressed as often as infeasible ones.

Default: 100%

horizon -- a Horizon\_Date field of model Problem\_Set

Obsolete! This field has been renamed 'tence', to prevent confusion with the new Horizon model. Please use 'tence' instead. 'horizon' will be removed in a later release. Default: oo\_future

owner -- a Strategy field of model Problem\_Set

Properties: Export-Only Field

Models	Strategy_Change Model
--------	-----------------------

### 5.4.2 Strategy\_Change Model

#### Strategy\_Change -- a submodel of model Strategy

Defines the relative 'focus' that should be given to each Change\_Category that can be performed to resolve Problems. For example, to avoid making anything later, you can specify that MOVE\_OUT is disallowed (0% focus).

Change\_Category's that do not appear in this List get the 'default\_change\_focus'.

The model has selectors:  
change\_category.

The Strategy\_Change model has fields that references these models :  
Strategy.

These models have a field that is a Strategy\_Change model :  
Strategy.

The key field for this model is change\_category

change\_category -- *an extension selector of model Strategy\_Change*  
The Change\_Category that should be tried with 'focus' relative weighting. Some examples:  
UPSTREAM (resolve a Problem by adjusting an upstream Operation\_Plan),  
DOWNSTREAM (resolve a Problem by adjusting a downstream Operation\_Plan),  
MOVE\_IN (move an Operation\_Plan to an earlier Date),  
MOVE\_OUT (move an Operation\_Plan to a later Date),  
RESIZE\_MORE (increase the Quantity of an Operation\_Plan),  
RESIZE\_LESS (decrease the Quantity of an Operation\_Plan),  
USE\_ALTERNATE\_OPERATION (use alternate Operations), or  
USE\_ALTERNATE\_RESOURCE (use alternate Resources).

By using different combinations of these, you can control what the Strategy will generally do to the plan. For example, by setting MOVE\_IN and/or RESIZE\_MORE to 100% focus and MOVE\_OUT and RESIZE\_LESS to 0%, you will tend to increase WIP, but will not increase lateness or shortness. By setting DOWNSTREAM to 0%, you will tend to move Buffer Problems upstream to the raw material Buffers.

Models	Strategy_Change Model
--------	-----------------------

For another example, suppose for a given strategy, a MOVE\_IN of 75 and a MOVE\_OUT of 25 are defined. These focus numbers stay the same, but they are used by resolvers along with the current state of the plan. For instance, the resource problem resolvers choose whether to move in, out, or off. They prefer moved to open space closest to the current plan. If open space before the start of the current operation plan is two weeks away, the resolvers weight more heavily the move in over the move out.

The RESIZE\_LESS and RESIZE\_MORE change\_categories should be thought of in terms of Operation\_Plan quantity, not Flow\_Plan quantity. For example, consider a Flow\_Plan consuming 10 from a buffer, which would appear as -10 in the editor. A RESIZE\_MORE applied to that Flow\_Plan would change it to -20 rather than -5.

An additional strategy change\_category that can affect buffer plan resolution is USE\_ALTERNATE\_OPERATION. As with other strategy focus change\_categories, the focus can vary. If set to zero, RHYTHM does not use alternate operations. Although changing to an alternate operation cannot always help solve buffer problems, there are situations where it can. An example is transportation, where moving from ground to air transportation may solve a problem related to receiving materials when needed.

Default: None -- this is a key field  
Extensions:  
MOVE\_IN, MOVE\_OUT, SPLIT, USE\_MORE, USE\_LESS,  
USE\_ALTERNATE\_OPERATION, USE\_ALTERNATE\_RESOURCE,  
USE\_EFFECTIVE\_ALTERNATE, INCREASE\_CAPACITY,  
DECREASE\_CAPACITY, RESIZE\_MORE, RESIZE\_LESS, UPSTREAM,  
DOWNSTREAM.

focus -- *a Percentage field of model Strategy\_Change*  
A relative weighting for how often the 'change\_category' should be tried while resolving Problems. "0%" indicates that the 'change\_category' should never be done. "100%" indicates that the 'change\_category' should be tried twice as often as "50%", but half as often as "200%". default is "100%", if the 'default\_change\_focus' is 0% and is "0%" otherwise.  
Default: 100% if 'default\_change\_focus' is 0%, 0% otherwise  
owner -- *a Strategy field of model Strategy\_Change*  
Properties: Export-Only Field

### 5.4.3 Strategy\_Lock Model

Strategy\_Lock -- a submodel of model Strategy

A list of specifications that describe what objects are to remain locked/unchanged for the duration of the strategy execution. For example, users can lock Operation\_Plan using Strategy\_Lock. The effects of multiple Strategy\_Locks are additive; should any Strategy\_Lock consider an Operation\_Plan locked; then it will be treated as such.

The model has selectors:  
spec.

The Strategy\_Lock model has fields that references these models :  
Strategy.

These models have a field that is a Strategy\_Lock model :  
Strategy.

The key field for this model is name

name - - a Symbol field of model Strategy\_Lock  
The name of this Strategy\_Lock.  
Default: None -- this is a key field

spec - - an extension selector of model Strategy\_Lock  
The 'spec' extension defines the additional fields required to specify locking criteria. For example, OPERATION\_PLAN\_RANK\_RANGE 'spec' extension specifies a range which will lock Operation\_Plan whose 'rank' lies within it. The OPERATION\_PLAN\_RANK\_EXPRESSION 'spec' extension allows user to use OIL expressions to determine which Operation\_Plan get locked.  
Default: OPERATION\_PLAN\_RANK\_RANGE  
Extensions:  
OPERATION\_PLAN\_RANK\_RANGE,  
OPERATION\_PLAN\_RANK\_EXPRESSION.

owner - - a Strategy field of model Strategy\_Lock

Properties: Export-Only Field

### 5.4.4 Strategy\_Goal Model

Strategy\_Goal -- a submodel of model Strategy

The goals of this strategy. This defines global goodness.

The model has selectors:  
goal.

The Strategy\_Goal model has fields that references these models :  
Strategy.

These models have a field that is a Strategy\_Goal model :  
Strategy, Active\_Goal.

The key field for this model is goal

goal - - an extension selector of model Strategy\_Goal  
The goal to be given 'focus' emphasis in the comparison of planning alternatives. The extension will add fields that define how to convert the absolute measure for the plan to a generic number that is numerically comparable to other goals of this Strategy.  
Default: None -- this is a key field  
Extensions:  
FEASIBILITY, MINIMIZE\_PROBLEM\_COUNT, MINIMIZE\_PROBLEMS, MINIMIZE\_LATENCY, WEIGHTED\_LATENCY, MINIMIZE\_SHORTNESS, WEIGHTED\_SHORTNESS, MINIMIZE\_COST, MAXIMIZE\_PROFIT, MAXIMIZE\_REVENUE.

focus\_to\_target - - a Percentage field of model Strategy\_Goal  
The focus (emphasis, weighting) on achieving the target for the 'goal' relative to other goals.  
Default: 100%

focus\_beyond\_target - - a Percentage field of model Strategy\_Goal  
The focus (emphasis, weighting) on achieving values over the target for the 'goal' relative to other goals.  
Default: 0%

owner - - a Strategy field of model Strategy\_Goal

Properties: Export-Only Field

5.4.5 execution submodels of model Strategy

5.4.6 Ordered\_Sub\_Strategy Model

Ordered\_Sub\_Strategy -- a submodel of model Strategy

An Ordered\_Sub\_Strategy that will be performed in particular sequence as part of the 'owner' Strategy.

The Ordered\_Sub\_Strategy model has fields that references these models :

Strategy.

These models have a field that is a Ordered\_Sub\_Strategy model :  
SEQUENCE\_RUN\_ONCE, SEQUENCE\_RUN\_MULTIPLE,  
ORDERED\_RESOLVES, SEQUENCE\_AFTER\_EACH\_RUN.

The key field for this model is sequence

sequence -- a Number field of model Ordered\_Sub\_Strategy  
The lower the Number, the earlier the Strategy is performed. No two  
Ordered\_Sub\_Strategy's of a Strategy may have the same 'sequence' number.  
Default: None -- this is a key field

strategy -- a Strategy field of model Ordered\_Sub\_Strategy  
A Strategy to be performed on the subset of Problems selected by the 'owner' Strategy.  
Default: (unspecified)

owner -- a Strategy field of model Ordered\_Sub\_Strategy

Properties: Export-Only Field

5.4.7 Ranked\_Sub\_Strategy Model

Ranked\_Sub\_Strategy -- a submodel of model Strategy

A Ranked\_Sub\_Strategy that can be selected and performed by the 'owner' Strategy.

The Ranked\_Sub\_Strategy model has fields that references these models :  
Strategy.

These models have a field that is a Ranked\_Sub\_Strategy model :  
PROPORTIONAL\_RESOLVES.

The key field for this model is strategy

rank -- a Percentage field of model Ranked\_Sub\_Strategy  
A Percentage that indicates how much focus will be given to this Sub\_Strategy. The  
greater the Percentage, the more time will be spent performing this Sub\_Strategy.  
Default: 100%

strategy -- a Strategy field of model Ranked\_Sub\_Strategy  
A Strategy to be performed on the subset of Problems selected by the 'owner' Strategy.  
Default: None -- this is a key field

owner -- a Strategy field of model Ranked\_Sub\_Strategy

Properties: Export-Only Field

Models	Unit Model
--------	------------

### 5.5 Unit Model

Unit -- *an independent (top-level) model*

A Unit defines the size (Quantity) of one unit of an Item, Product, or Operation. Effectively, it relates "unit of Item" to a "unit of Measure". For instance, one unit of Paint304 may be "1 gal", "2 kg", and/or "\$4.25".

The Unit model has these submodels :  
Unit\_Quantity.

The Unit model has fields that references these models :  
Unit\_Quantity.

These models have a field that is a Unit model :  
Operation, Buffer, Product\_Root, Item, Product\_Group, Unit\_Quantity.

The key field for this model is preferred\_measure

preferred\_measure -- *a Measure field of model Unit*  
The preferred Measure to be used when displaying Quantities of this Unit. This Measure must be specified in the 'quantities', or be the "unitless" Measure which is simply the number of units.

For example, if one unit of Paint304 is "1 gal" and "2 kg", then this could be volume, mass, or unitless; giving "10 gal", "20 kg", or "10", respectively, for ten units of Paint304.

Default: None -- this is a key field

discrete -- *a Logical field of model Unit*  
If "true", then this (the one that is measured) can only exist in whole units. In that case, the Quantity used will always be rounded up to a whole number of units of this. Note that a whole unit of this may not be a whole unit of Measure.  
Default: false

quantities -- *a list of Unit\_Quantity submodels of model Unit*  
Specifies zero or more measurements of this Unit. For example, this Unit may be "10kg" in mass. This Unit may also be "16 cm" in height, "4 liters" in volume, and "\$120" in money. Each of those four would be specified as a separate Unit\_Quantity. The default Measure used for this Unit is specified in the 'preferred\_measure' field.

Models	Unit Model
--------	------------

Note that all Units may be expressed in the "unitless" Measure, which is simply the number of those Units.

convert (Measure\_Unit, Measure) -- *a Quantity field of model Unit*  
Returns the Quantity of this Unit with the given unit of Measure that is equivalent to the given Quantity. Both Measures must be defined in 'quantities', or be "unitless" meaning that number of this Unit. If 'discrete' is "true", then this will round up to a whole number of these Units (which is not necessarily a whole number of the requested unit of Measure).  
Properties: Export-Only Field

Models	Unit_Quantity Model
--------	---------------------

### 5.5.1 Unit\_Quantity Model

Unit\_Quantity -- *a submodel of model Unit*

Specifies zero or more measurements of this Unit. For example, this Unit may be "10kg" in mass. This Unit may also be "16 cm" in height, "4 liters" in volume, and "\$120" in money. Each of those four would be specified as a separate Unit\_Quantity. The default Measure used for this Unit is specified in the 'preferred\_measure' field.

Note that all Units may be expressed in the "unitless" Measure, which is simply the number of those Units.

The Unit\_Quantity model has fields that references these models :

These models have a field that is a Unit\_Quantity model :

The key field for this model is quantity

quantity - - - *a Quantity field of model Unit\_Quantity*

Specifies a measurement of this Unit. This is the Quantity of that Measure that is equivalent to this Unit. Each Unit\_Quantity must have a unique Measure (e.g., two different measurements of a Unit cannot have volume Measure).

Default: None -- this is a key field

owner - - - *a Unit field of model Unit\_Quantity*

Properties: Export-Only Field

Models	Profile_Number Model
--------	----------------------

### 5.6 Profile\_Number Model

Profile\_Number -- *an independent (top-level) model*

A List(Profile\_Number) is used to profile a Number over time. A Profile\_Number defines for a Number a new 'value' at 'date' and new 'rate' continuing from 'date' until the next Profile\_Number. For example, at "95-07-23" the Number may change to "1200" and the rate may change to "50/hr". It will continue at "50/hr" until the 'date' of the next Profile\_Number.

Note that the new 'value' may be a discrete jump at 'date', independent of the 'rate's before and after that 'date'. Thus, a Profile\_Number may correspond to a discrete jump, but the rate may remain the same. Similarly, a Profile\_Number may correspond to a 'rate' change on a 'date' where there is no discrete jump in 'value'.

The key field for this model is date

date - - - *a Date field of model Profile\_Number*

The Date at which this change in either 'value' or 'rate' occurs.

Default: None -- this is a key field

Properties: Export-Only Field

value - - - *a Number field of model Profile\_Number*

The new Number value at 'date'.

Properties: Export-Only Field

rate - - - *a Quantity field of model Profile\_Number*

The new Number rate of change (per unit Time) which continues until the next Profile\_Number.

Properties: Export-Only Field



<i>Models</i>	<i>Profile_Percentage Model</i>
---------------	---------------------------------

### 5.7 Profile\_Percentage Model

**Profile\_Percentage** -- *an independent (top-level) model*

A List(Profile\_Percentage) is used to profile a Percentage over time. A Profile\_Percentage defines for a Percentage a new 'value' at 'date' and new 'rate' continuing from 'date' until the next Profile\_Percentage. For example, at "95-07-23" the Percentage may change to "88%" and the rate may change to "2%/hr". It will continue at "2%/hr" until the 'date' of the next Profile\_Percentage.

Note that the new 'value' may be a discrete jump at 'date', independent of the 'rate's before and after that 'date'. Thus, a Profile\_Percentage may correspond to a discrete jump, but the rate may remain the same. Similarly, a Profile\_Percentage may correspond to a 'rate' change on a 'date' where there is no discrete jump in 'value'.

The key field for this model is date

**date** -- *a Date field of model Profile\_Percentage*  
 The Date at which this change in either 'value' or 'rate' occurs.  
 Default: None -- this is a key field  
 Properties: Export-Only Field

**value** -- *a Percentage field of model Profile\_Percentage*  
 The new Percentage value at 'date'.  
 Properties: Export-Only Field

**rate** -- *a Quantity field of model Profile\_Percentage*  
 The new Percentage rate of change (per unit Time) which continues until the next Profile\_Percentage.  
 Properties: Export-Only Field

<i>Models</i>	<i>Profile_Quantity Model</i>
---------------	-------------------------------

### 5.8 Profile\_Quantity Model

**Profile\_Quantity** -- *an independent (top-level) model*

A List(Profile\_Quantity) is used to profile a Quantity over time. A Profile\_Quantity defines for a Quantity a new 'value' at 'date' and new 'rate' continuing from 'date' until the next Profile\_Quantity. For example, at "95-07-23" the Quantity may change to "1200 kg" and the rate may change to "50kg/hr". It will continue at "50kg/hr" until the 'date' of the next Profile\_Quantity.

Note that the new 'value' may be a discrete jump at 'date', independent of the 'rate's before and after that 'date'. Thus, a Profile\_Quantity may correspond to a discrete jump, but the rate may remain the same. Similarly, a Profile\_Quantity may correspond to a 'rate' change on a 'date' where there is no discrete jump in 'value'.

The key field for this model is date

**date** -- *a Date field of model Profile\_Quantity*  
 The Date at which this change in either 'value' or 'rate' occurs.  
 Default: None -- this is a key field  
 Properties: Export-Only Field

**value** -- *a Quantity field of model Profile\_Quantity*  
 The new Quantity value at 'date'.  
 Properties: Export-Only Field

**rate** -- *a Quantity field of model Profile\_Quantity*  
 The new Quantity rate of change (in 1/s units) which continues until the next Profile\_Quantity.  
 Properties: Export-Only Field

Models	Box Model
--------	-----------

### 5.9 Box Model

Box -- *an independent (top-level) model*

A Box defines a 2-dimensional box (x, y, width, height). All coordinates are floats, which can be in any unit you want, because they're all relative to one another.

These models have a field that is a Box model :  
Location.

The key field for this model is specification

specification -- *a Computed\_String field of model Box*  
String representation of the box formatted as: "1x2+3+4" Where '1' is the width, '2' is the height, '3' is the x coordinate, and '4' is the y.  
Default: None -- this is a key field

x -- *a Number field of model Box*  
Horizontal position of the upper left corner of the box.  
Default: 0

y -- *a Number field of model Box*  
Vertical position of the upper left corner of the box.  
Default: 0

width -- *a Number field of model Box*  
Width of the box  
Default: 0

height -- *a Number field of model Box*  
Height of the box  
Default: 0

area -- *a Number field of model Box*  
Area of the box  
Properties: Export-Only Field

mid\_x -- *a Number field of model Box*  
Horizontal position of the center of the box  
Default: 0

Models	Box Model
--------	-----------

mid\_y -- *a Number field of model Box*  
Vertical position of the center of the box  
Default: 0

empty -- *a Logical field of model Box*  
TRUE if the box has zero area  
Properties: Export-Only Field

unspecified -- *a Logical field of model Box*  
TRUE if the box is unspecified  
Properties: Export-Only Field

Models	Calendar Model
--------	----------------

### 5.10 Calendar Model

Calendar -- *an independent (top-level) model*

Calendar models daily patterns (e.g., 08:00 to 16:00 each day) that themselves may repeat in patterns related to calendar years, months, weeks, or days. For example, the 08:00 to 16:00 pattern may repeat for all weekdays (M-F), or for all third Tuesdays of each month.

Each pattern can have an associated value. For example, a Number representing the efficiency of 80% may be associated with 08:00 to 16:00 on weekdays, or a Symbol that is a setup name may be associated with the second week of each month. Each field that is of type Calendar will demand a particular 'entry\_value' (the type of the value(s) in each of its entries). It is an error to use a Calendar with the wrong 'entry\_value'.

Each Calendar\_Entry in a Calendar specifies a value applicable for a certain pattern of Dates. Those Date patterns may overlap. In that case, the Calendar\_Entry with the higher 'rank' is used. That allows you to specify 08:00 to 16:00 on all weekdays (very simple to do) and then specify an overriding Calendar\_Entry for Thanksgiving and the rest of your holidays. That is much easier than if you had to put the holiday exceptions into each and every Calendar\_Entry.

Furthermore, one Calendar can be built up from other Calendars. The other Calendars' 'entries' are brought into the Calendar, with their 'rank's adjusted according to an Expression. In that way, several different Calendars could be defined, and each of those could use the same "holiday" Calendar to override the holidays.

There is one special uneditable Calendar named [unspecified] that has no 'entries' and no 'sub\_calendars', nor can any entries, or sub\_calendars be added to the [unspecified] calendar. The [unspecified] calendar has no values.

The model has selectors:  
entry\_value.

The Calendar model has these submodels :  
Calendar\_Entry, Sub\_Calendar.

The Calendar model has fields that references these models :  
Calendar\_Entry, Sub\_Calendar.

These models have a field that is a Calendar model :

Models	Calendar Model
--------	----------------

Calendar\_Entry, Sub\_Calendar, Effective\_Calendar\_Operation, PRODUCE\_YIELD\_CALENDAR, PRODUCE\_YIELD\_RAMP\_CALENDAR, DELAY\_ONLY\_CALENDAR, BASIC\_CALENDARS, SETUP\_CALENDAR, SKILL\_CALENDAR, TRANSIT\_CALENDAR, CALENDAR, RAMP\_CALENDAR, SETUP\_CALENDAR, BLOCK\_CALENDAR, CAL- ENDAR, CALENDAR, CALENDAR\_COUNT, CALENDAR\_QUANTITY, CALENDAR\_RATE, CALENDAR, CALENDAR, PRODUCING\_FLOW\_CALENDAR, PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK, SUPPLY\_CALENDAR, ON\_HAND\_CALENDAR, ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK, FLOW\_LIMIT\_CALENDAR, FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK, CALENDAR, Calendar\_Plan.

The key field for this model is name  
This model may be extended with user-defined fields.

name -- *a Symbol field of model Calendar*  
The name of this Calendar.  
Default: None -- this is a key field

description -- *a String field of model Calendar*  
A description of this Calendar.  
Default: none

entry\_value -- *an extension selector of model Calendar*  
This extension selector dictates the 'value' extension of all Calendar\_Entry's within this Calendar, which specifies the value(s) that are held in each entry.

For example, NUMBER has a single Number in each entry, such as might be used to represent a Calendar of Efficiencies. QUANTITY has a single Quantity in each entry, as might be used to represent replenishment supply to a Buffer.  
NUMBER\_QUANTITY has a Number and a Quantity in each entry. SYMBOL has a Symbol in each Entry, as might be used in a Calendar of allowed setups.  
Default: UNSPECIFIED  
Extensions:  
UNSPECIFIED, NUMBER, QUANTITY, NUMBER\_QUANTITY, SYMBOL, TIME.

entries -- *a list of Calendar\_Entry submodels of model Calendar*  
The individual entries that make up this Calendar.

**entry (Date)** - - *a Calendar\_Entry field of model Calendar*

Returns the Calendar\_Entry (from 'entries' or sub\_calendars) that has the highest rank at the given Date.

Properties: Export-Only Field

**dates (Date\_Range)** - - *a list(Date) field of model Calendar*

This returns a list(Date) at which the 'entry' of the Calendar changes during the specified Date\_Range. This can be used in conjunction with 'entry' to compute the list of Calendar\_Entry's that are in effect during that Date\_Range.

Properties: Export-Only Field

**sub\_calendars** - - *a list of Sub\_Calendar submodels of model Calendar*

The Calendars upon which this one is built. The 'entries' in each of the 'sub\_calendars' are incorporated into this Calendar, with 'rank' modified according to 'rank\_expression'.

As an example of the usage, consider various Calendars created for various weekday shift patterns. Each of those are most easily specified by giving the shift hours and applying that to every weekday (Monday through Friday). However, all of those Calendars need to account for holidays. Although it is a Thursday, most of those shifts will probably not run on Thanksgiving Day. Rather than duplicate the holiday entries in each of those shift Calendars, one Calendar can be created with the holidays, and all the other Calendars can simply specify the holiday Calendar as a sub\_calendar. By making the holiday Calendar\_Entry's 'rank' higher than the shift Calendar\_Entry's, the holiday entries will be used instead on the days they apply.

ALL CALENDARS IN A CALENDAR/SUB\_CALENDAR HIERARCHY MUST HAVE THE SAME 'entry\_value' extension.

### 5.10.1 Calendar\_Entry Model

**Calendar\_Entry** - - *a submodel of model Calendar*

Each Calendar\_Entry represents a particular value that is valid during a time range each day that it applies, and the 'day\_pattern' of days that it does apply during the Date\_Range in which it is 'effective'. For example, a Calendar\_Entry could be effective during 1995 that specifies 80% from 8:00 to 16:00 on every weekday.

Further, each Calendar\_Entry can specify a 'rank' which is used to resolve overlapping Calendar\_Entry's. For example, another Calendar\_Entry could specify 0% from 0:00 to 24:00 on December 25. If December 25 is a weekday in 1995, then it and the 80% entry above would both apply. In that case, the entry with the higher 'rank' would override.

Finally, each Calendar\_Entry can specify a 'charge'. Such an entry is not used by default, but is instead an alternative. Using that alternative has an associated cost, as modeled by the 'charge'.

The model has selectors:  
value, day\_pattern.

The Calendar\_Entry model has fields that references these models:  
Calendar.

These models have a field that is a Calendar\_Entry model:  
Calendar, Calendar\_Plan.

The key field for this model is name  
This model may be extended with user-defined fields.

name - - *a Symbol field of model Calendar\_Entry*

The name of this Calendar\_Entry. Calendar\_Entry names must be unique within the scope the owning calendar.

Default: None - - this is a key field

value - - *an extension selector of model Calendar\_Entry*

This specifies the type of value(s) held by this Calendar\_Entry. This is the same as the 'owner' Calendar's 'entry\_value' extension. It is not settable here.

For example, NUMBER has a single Number in each entry, such as might be used to represent a Calendar of Efficiencies. QUANTITY has a single Quantity in each entry, as might be used to represent replenishment supply to a Buffer.

NUMBER\_QUANTITY has a Number and a Quantity in each entry. SYMBOL has a Symbol in each Entry, as might be used in a Calendar of allowed setups. TIME has a Time in each Entry.

Default: NUMBER

Properties: Export-Only Field

Extensions:

NUMBER, QUANTITY, NUMBER\_QUANTITY, SYMBOL, TIME.

description - - *a String field of model Calendar\_Entry*

Comment about this particular entry (e.g., "Joe is on vacation")

Default: the empty String

effective - - *a Date\_Range field of model Calendar\_Entry*

The period during which this Calendar\_Entry may be applicable. The Calendar\_Entry may not be active before, nor after, the effective period. The effective period does not specify the time a Calendar\_Entry will be effective, it specifies the period of time during which a Calendar\_Entry may be effective. Whether, or not a Calendar\_Entry is actually applicable on a given day depends on the 'day\_pattern' that is used, the 'daily\_start' and 'daily\_end' times, and the Calendar\_Entry's rank.

For example, assume a Calendar entry uses an EVERYDAY day pattern, has an effective period of 97-01-10 00:00 / 97-01-15 00:00, a daily\_start of 18:00, and a daily\_end of 26:00, or 02:00. The entry cannot start on 97-01-09 18:00 because that time is outside the effective period. The soonest it can be started is 97-01-10 00:00, which is the first second of the effective period. Similarly, The calendar entry cannot end at 97-01-16 02:00, because that time is outside the effective period. The longest it can run is 97-01-15 24:00.

The day\_pattern, daily\_start, and daily\_end define a repeating pattern. Only that part of the pattern in the effective period is visible. In essence this is an logical AND operation between the repeating pattern and the effective period.

'effective.start' specifies the earliest time a Calendar\_Entry may be applicable.

'effective.end' specifies the time a Calendar\_Entry can no longer be applicable. The second prior to 'effective.end' is the last second the Calendar\_Entry may be applicable.

Default: forever

daily\_start - - *a Time field of model Calendar\_Entry*

For each day on which a Calendar\_Entry is applicable, 'daily\_start' is the time that the 'value' specified goes into effect.

If 'daily\_start' is < 0, it specifies a start time on a prior day. If 'daily\_start' is >= 24:00, it specifies a start time on a subsequent day.

See documentation in daily\_end for more details and examples.

Default: 00:00:00

daily\_end - - *a Time field of model Calendar\_Entry*

For each day on which a Calendar\_Entry is applicable, 'daily\_end' is the time that the 'value' is no longer in effect.

If 'daily\_end' is < 0, it specifies an end time on a prior day. If 'daily\_end' is >= 24:00, it specifies an end time on a subsequent day.

If 'daily\_end' is less than 'daily\_start', it specifies an end time that will occur after 'daily\_start'. For example, if 'daily\_start' is "18:00" and 'daily\_end' is "02:00", then the actual times are "18:00" and "26:00", or from 6:00pm until 2:00am on the next day.

Both 'daily\_start' and 'daily\_end' are Time values. A time < "00:00" specifies a 'daily\_start', or 'daily\_end', on a prior day. A time >= "24:00" specifies a 'daily\_start', or 'daily\_end', on a subsequent day. This allows a Calendar\_Entry to specify events which precede, or follow, the day on which a Calendar\_Entry is applicable.

Examples:

An 8-hour work shift that begins at 10:00pm can be specified using a 'daily\_start' "22:00", and a 'daily\_end' of "30:00", or "06:00".

To specify a Christmas holiday from 97-12-24/97-12-25, a Calendar\_Entry could be specified that is applicable on 12-25 which has a 'daily\_start' of "-24hr" and a 'daily\_end' of "24hr". The actual start is 97-12-24 00:00 and the actual end is 97-12-25 24:00.

To specify a Thanksgiving holiday on the 4th Thursday in November, and the following day, a Calendar\_Entry that is applicable on the 4th Thursday could have a 'daily\_start' of "00:00" and a 'daily\_end' of "48:00", which ends on Friday at "24:00".

In both the examples above, the day before Christmas and the day after Thanksgiving could have been specified using separate calendar entries. One using a DAY\_OF\_MONTH day pattern for 12-24, and the other using a DAY\_OF\_WEEK\_OF\_MONTH day pattern for the fourth Friday in November. However, this requires two separate calendar entries when only one is needed.

As a final example, consider the 3-day weekend of Labor Day which falls on the first Monday in September. This can be specified using the DAY\_OF\_WEEK\_OF\_MONTH day pattern for the first Monday in September, with a 'daily\_start' of "-48hr" and a 'daily\_end' of 24hr. This is much easier than adding two more entries for the days preceding the first Monday in September, and it even works when Labor Day falls on September 1.  
Default: 24:00:00

day\_pattern - - an extension selector of model Calendar\_Entry

This and the related fields define the days that this Calendar\_Entry applies. For example, it may specify that it applies on weekdays, on M,W,F, on the third Tuesday of any month, on the fourth Thursday in November, on the 25th day of December, on the 24th of December only if it is a Thursday, or just on 91-07-23.

Default: EVERYDAY

Extensions:

EVERYDAY, EVERY\_N\_DAYS, WEEKDAYS, WEEKENDS,  
DAYS\_OF\_WEEK, DAY\_OF\_MONTH, DAY\_OF\_WEEK\_OF\_MONTH,  
DAY\_OF\_LAST\_WEEK\_OF\_MONTH, DAY\_OF\_YEAR, YEARLY.

rank - - a Number field of model Calendar\_Entry

For days on which multiple Calendar\_Entry's apply, the entry with the highest rank is used. Thus, this field allows an entry for the 25th day of December to override an entry for weekdays (normal workshift overridden by a holiday).

Default: 1

charge - - a Quantity field of model Calendar\_Entry

If this is non-zero, then there is a cost associated with using this Calendar\_Entry. For example, running overtime will have a charge. Coupled with the 'rank' field, you could for example set up an entry that overrides a holiday (high rank), but has a non-zero charge, and thus is only used when necessary.

Default: 0

owner - - a Calendar field of model Calendar\_Entry

Properties: Export-Only Field

5.10.2 Sub\_Calendar Model

Sub\_Calendar - - a submodel of model Calendar

The Calendars upon which this one is built. The 'entries' in each of the 'sub\_calendars' are incorporated into this Calendar, with 'rank' modified according to 'rank\_expression'.

As an example of the usage, consider various Calendars created for various weekday shift patterns. Each of those are most easily specified by giving the shift hours and applying that to every weekday (Monday through Friday). However, all of those Calendars need to account for holidays. Although it is a Thursday, most of those shifts will probably not run on Thanksgiving Day. Rather than duplicate the holiday entries in each of those shift Calendars, one Calendar can be created with the holidays, and all the other Calendars can simply specify the holiday Calendar as a sub\_calendar. By making the holiday Calendar\_Entry's 'rank' higher than the shift Calendar\_Entry's, the holiday entries will be used instead on the days they apply.

ALL CALENDARS IN A CALENDAR/SUB\_CALENDAR HIERARCHY MUST HAVE THE SAME 'entry\_value' extension.

The Sub\_Calendar model has fields that references these models :

Calendar.

These models have a field that is a Sub\_Calendar model :

Calendar.

The key field for this model is calendar

calendar - - a Calendar field of model Sub\_Calendar

Each of the 'entries' of 'calendar' and its subcalendars will be incorporated into the 'owner' Calendar. This allows, for example, a holiday Calendar to be defined once with high-ranking entries, and then reused by each of the other Calendars.

Default: None - - this is a key field

rank\_expression - - a Expression field of model Sub\_Calendar

For each Calendar\_Entry included from 'calendar', the rank is reset to the value returned from this Expression. The variable 'rank' is defined for this Expression to be the original 'rank' of that Calendar\_Entry.

This allows the ranking schemes of two separately defined Calendars to be adjusted to fit together, and allows them to be incorporated into the ranking scheme of the 'owner' Calendar.

Default: rank

owner - - a Calendar field of model Sub\_Calendar

Properties: Export-Only Field

### 5.11 Calendar\_Plan Model

Calendar\_Plan -- an independent (top-level) model

Calendar\_Plan has two primary purposes:

- allow you to iterate through the Dateline (the list of resulting entries)
- allow you to select entries with cost that you want to use

The model has selectors:

entry\_value.

The Calendar\_Plan model has fields that references these models :  
Calendar, Calendar\_Entry.

The key field for this model is calendar

calendar - - a Calendar field of model Calendar\_Plan  
The Calendar that defines this Calendar\_Plan.

This is not directly settable. It is generally specified by a model associated with the model that provides the Calendar\_Plan field. See the documentation for the Calendar\_Plan field of interest.

Default: None -- this is a key field

Properties: Export-Only Field

horizon - - a Date\_Range field of model Calendar\_Plan

The range of Dates for which the Calendar 'entries' are used to determine the values of this Calendar\_Plan. Before and after this 'horizon', a single value will be used, as dictated by the model which contains the Calendar\_Plan.

This is not directly settable. It is generally specified by a model associated with the model that provides the Calendar\_Plan field. See the documentation for the Calendar\_Plan field of interest.

Properties: Export-Only Field

entry\_value - - an extension selector of model Calendar\_Plan

This 'entry\_value' extension defines the fields which specify the value to be used before and after the 'horizon'.

Models	Calendar_Plan Model
--------	---------------------

Note that the 'calendar' must have the same 'entry\_value' as this, or have UNSPECIFIED 'entry\_value'.

This is not settable. It is dictated by the model that provides the Calendar\_Plan field. See the documentation for the Calendar\_Plan field of interest.

Properties: Export-Only Field

Extensions:

NUMBER, QUANTITY, NUMBER\_QUANTITY, SYMBOL, TIME.

entry (Date) -- a Calendar\_Entry field of model Calendar\_Plan

Returns the Calendar\_Entry (from 'entries' or 'sub\_calendars') that has the highest rank at the given Date.

Properties: Export-Only Field

dates (Date\_Range) -- a List[Date] field of model Calendar\_Plan

This returns a List[Date] at which the 'entry' of the Calendar changes during the specified Date\_Range. This can be used in conjunction with 'entry' to compute the list of Calendar\_Entry's that are in effect during that Date\_Range.

Properties: Export-Only Field

Control Model	Calendar_Plan Model
---------------	---------------------

### 6 Control Model

Control -- an independent (top-level) model



7 Connection Model

Connection -- an independent (top-level) model

These models have a field that is a Connection model :  
User.

7.1 User Model

User -- an independent (top-level) model

A User is a model of a human user of the system, or a group ("family") of users. User-specific settings are provided by this model. In particular, User security settings and User-specific directories for Reports, Layouts, Worksheets, and Styles are specified here. Also User-specific Event bindings and User-specific Command definitions are provided by this model.

The User model has these submodels :  
Data\_Directory, Report\_Directory, Report, Layout, Worksheet, Style, Format, Domain.

The User model has fields that references these models :  
User, Data\_Directory, Report\_Directory, Layout, Worksheet, Style, Format, Domain, Connection.

These models have a field that is a User model :  
User, Data\_Directory, Report\_Directory, Format.

The key field for this model is name  
This model may be extended with user-defined fields.

name -- a Symbol field of model User  
The User's name or system account, by which the security is enforced.  
Default: None -- this is a key field

full\_name -- a String field of model User  
The User's full name.  
Default: 'name'

responsibility -- a String field of model User  
The User's responsibility or position.  
Default: none

remark -- a String field of model User  
Often used to describe what a User is doing, where a User is, or other useful cooperative information.  
Default: none

Connection Model	User Model
------------------	------------

organization - - *a User field of model User*  
 The User organization to which this User belongs.  
 Default: none

members - - *a List(User) field of model User*  
 If this User is an organization, then this List contains each User that specifies this User as its 'organization'.  
 Properties: Export-Only Field

member\_of (User) - - *a Logical field of model User*  
 Returns "true" if this User or any of its 'organization's is the parameter User. More precisely, if this User is the parameter User, then it returns "true"; otherwise, if this User's 'organization' is nonexistent, then it returns "false"; otherwise, it returns 'organization.member\_of(parameter)'.  
 Properties: Export-Only Field

super\_user - - *a Logical field of model User*  
 The user who starts the engine is the "super User" and has full privileges. If this is "true", then this User is the "super User".  
 Properties: Export-Only Field

data\_directories - - *a list of Data\_Directory submodels of model User*  
 The directories where data Worksheet definitions can be found and saved. Typically, text files to be imported or exported also reside here.

current\_data\_directory - - *a Pathname field of model User*  
 The current data worksheet directory  
 Default:

load\_data\_layouts (String) - - *a Void field of model User*  
 loads the import files in the given directory. These can then be obtained from the import\_files field.  
 Properties: command=True Export-Only Field

inc\_new\_data\_layout (String, String) - - *a Void field of model User*  
 create and inc a new import file  
 Properties: command=True Export-Only Field

report\_directories - - *a list of Report\_Directory submodels of model User*  
 The directories where Report, Layout, Worksheet, Format and Style definitions can be found and saved.

Connection Model	User Model
------------------	------------

reports - - *a list of Report submodels of model User*  
 The Layouts defined in this User's 'report\_directories'. Layouts define the layout of information computed in a Worksheet. The same Worksheet can be displayed in numerous different tabular forms, as a bar chart, a line chart, a Gantt chart, and so on.  
 The Reports defined in this Users' 'report\_directories'. Reports can be used interactively (in GUI windows), printed to paper, or can generate files.

layouts - - *a list of Layout submodels of model User*  
 The Layouts defined in this User's 'report\_directories'. Layouts define the layout of information computed in a Worksheet. The same Worksheet can be displayed in numerous different tabular forms, as a bar chart, a line chart, a Gantt chart, and so on.

worksheets - - *a list of Worksheet submodels of model User*  
 The Worksheets defined in this User's 'report\_directories'. Worksheets are the computational elements of Reports.

styles - - *a list of Style submodels of model User*  
 The Styles defined in this User's 'report\_directories'. Styles define the "look" of Controls and Layouts. They specify colors, borders, alignment, and fonts.

formats - - *a list of Format submodels of model User*  
 The Formats defined in this User's 'report\_directories'. Formats define how values are converted to and from Strings.

domains - - *a list of Domain submodels of model User*  
 The report domains defined in this User's main report.

load\_files - - *a Logical field of model User*  
 Loads the report files, layout files, and worksheet files owned by this user. Returns "true" if the load appears successful. If "false", the User may lack permissions or there may be some other file system problem.  
 Properties: command=True Export-Only Field

connections - - *a List(Connection) field of model User*  
 Return the list of connections used by this user  
 Properties: Export-Only Field

active\_ui - - *a Connection field of model User*  
 Return the last connection created for this user

Connection Model	User Model
Properties: Export-Only Field	
language - - a Symbol field of model User	
The Language preferred by this User. All text that has been translated to the specified language will be displayed translated.	
Default: English	

Connection Model	Report_Directory Model
------------------	------------------------

7.1.1 Report\_Directory Model

Report\_Directory -- a submodel of model User

The directories where Report, Layout, Worksheet, Format and Style definitions can be found and saved.

The Report\_Directory model has fields that references these models :  
User.

These models have a field that is a Report\_Directory model :  
User.

The key field for this model is sequence

sequence - - a Number field of model Report\_Directory  
The smaller the number, the earlier it is searched.  
Default: None -- this is a key field

directory - - a Pathname field of model Report\_Directory  
The file system pathname for a directory that contains Report, Layout, Worksheet, and/or Style definitions. If this is set, then 'include' will be set to [unspecified].  
Default: none

include - - a User field of model Report\_Directory  
Use the include directories of the specified user. If this is set, then 'directory' will be unspecified.  
Default: [unspecified]

description - - a String field of model Report\_Directory  
A description of the kinds of Reports saved in this directory. In particular, whether or not this directory is shared with other users is important.  
Default: none

editable - - a Logical field of model Report\_Directory  
If "true" then the 'owner' User has permission to edit this directory. This cannot be set "true" if the User's system account does not have permission to modify the directory. It can be set "false" either way.  
Default: the native file system permissions.

owner - - a User field of model Report\_Directory

Connection Model	Report_Directory Model
------------------	------------------------

Properties: Export-Only Field

Connection Model	Report Model
------------------	--------------

7.1.2 Report Model

Report -- a submodel of model User

These models have a field that is a Report model :  
User.

Connection Model	Layout Model
------------------	--------------

7.1.3 Layout Model

Layout -- a submodel of model User

These models have a field that is a Layout model :  
User.

Connection Model	Worksheet Model
------------------	-----------------

7.1.4 Worksheet Model

Worksheet -- a submodel of model User

These models have a field that is a Worksheet model :  
User, Data\_Directory.

Connection Model	Style Model
------------------	-------------

### 7.1.5 Style Model

Style -- a submodel of model User

These models have a field that is a Style model :

User.

Connection Model	Format Model
------------------	--------------

### 7.1.6 Format Model

Format -- a submodel of model User

A Format defines how to format a value of Type 'type' into a String. It also define how to parse a String back into a value of Type 'type'. Formats are used primarily to effect the interactive display of values in Reports. For consistency, most Reports use the same named Formats; and that allows a User to override the definition of a named Format and affect the formatting consistently throughout the Reports.

The model has selectors:

spec.

The Format model has fields that references these models :

User.

These models have a field that is a Format model :

User.

The key field for this model is name\_type

name\_type -- a Symbol field of model Format

The name of this Format underscore the type for the format. Typically the base-name of the file system filename where this Format is saved. For example, the default date format will have a name\_type of "default\_date". The name field will be 'default'; and the 'spec' field will be 'date'.

Default: None -- this is a key field

name -- a Symbol field of model Format

The name of this format without the data-type appended to it. For example, if the Format.name\_type is 'short\_date\_range', then format.name will be 'short', and format.spec will be 'date\_range'.

Default: Name from the name\_type field

directory -- a Pathname field of model Format

The Pathname of the native file system directory where this Format's definition is saved.

Default: the Pathname of the directory where it was originally found

description -- a String field of model Format

A description of this Format: what it computes, how it can be used, where it is intended to be used.

Connection Model	Format Model
------------------	--------------

Default: none

editable -- a logical field of model Format

If "true", then this Format can be edited.

Properties: Export-Only Field

save, save (Pathname) -- a logical field of model Format

Saves this Format into 'directory'. Returns "true" if the save appears successful. If "false", the User may lack permissions or there may be some other file system problem.

Properties: Export-Only Field

handles (Type) -- a logical field of model Format

Returns "true" if this format can format and parse values of the specified Type.

Properties: Export-Only Field

spec -- an extension selector of model Format

Defines the type of values that can be formatted and the fields that will specify how to format those values into Strings. Those fields also affect the parsing of Strings into values.

Default: Void

Extensions:

Void, Logical, String, Symbol, Date, Date\_Range, Number, Percentage, Integer, Quantity, Quantity\_Range, Time, Restriction, Horizon\_Date, List.

owner -- a User field of model Format

Properties: Export-Only Field

Connection Model	Domain Model
------------------	--------------

### 7.1.7 Domain Model

Domain -- a submodel of model User

These models have a field that is a Domain model :  
User.

7.2 Model\_Type Model

Model\_Type -- *an independent (top-level) model*

A Model\_Type models a model. Each model can be used as a value of a particular Type, a Model\_Type, in Expressions. The Model\_Type model describes that particular Type and the Fields that it provides.

The primary uses for the Model\_Type model are to provide help documentation and to allow user-defined Fields to be added to extensible Model\_Types. Note that you can also add user-defined Model\_Types.

The Model\_Type model has these submodels :  
Field, Extension\_Selector.

The Model\_Type model has fields that references these models :  
Model\_Type, Field, Extension\_Selector.

These models have a field that is a Model\_Type model :  
Model\_Type, Field, Extension\_Selector.

The key field for this model is name  
This model may be extended with user-defined fields.

name -- *a Symbol field of model Model\_Type*  
The name of the Model\_Type, which is also a Type name in the Expression language. This is only editable if 'user\_defined' is "true".  
Default: None -- this is a key field

description -- *a String field of model Model\_Type*  
A description of the Model\_Type named 'name'. This is only editable if 'user\_defined' is "true".  
Default: none

owner\_model -- *a Model\_Type field of model Model\_Type*  
The Model\_Type of this Model\_Type's 'owner' field. This Model\_Type is a submodel of 'owner\_model'. Top-level Model\_Types will return the Void Model\_Type. This is only editable if 'user\_defined' is "true".  
Default: none  
Properties: Export-Only Field

extensible -- *a Logical field of model Model\_Type*

Returns "true" if user-defined Fields may be added to 'fields' of this Model\_Type. Some small, numerous models do not support the addition of user-defined fields to avoid the small overhead required (which becomes significant in large numbers). This is always "true" if 'user\_defined' is "true".  
Properties: Export-Only Field

access -- *a Expression field of model Model\_Type*

A logical expression that returns "true" if the current user can set the fields in this model. The 'access' Expression is passed '#' (equal to this model). The current user is available via the user 'worksheet' function.  
Default: True

values (Typed\_Value) -- *a List(Typed\_Value) field of model Model\_Type*

Given an instance of this model's owner, return the list of all . instances with that owner. This can be used to access the "instance" field of the model; to get all the values. Example usage:  
typed\_value.type.values(typed\_value.owner)  
Properties: Export-Only Field

fields -- *a list of Field submodels of model Model\_Type*

The Fields provided by this Model\_Type. Each is a function in the Expression language that can be called on a model of this Model\_Type. Fields can only be added if 'extensible' is "true".

extension\_selectors -- *a list of Extension\_Selector submodels of model*

Model\_Type  
Each of these can add a different extension to a model, effectively adding some new fields to that model.



7.2.1 Field Model

Field -- a submodel of model Model\_Type

A Field is a function that takes a model as a first argument. It returns one attribute of or the result of an analysis on that model. The Model\_Type of that model determines what Fields are defined.

User-defined Fields can be added to many of the Model\_Types. Only user-defined Fields may be removed. Note, however, that all data stored in that Field or a model disappears when it is removed.

The model has selectors:  
field\_type.

The Field model has fields that references these models :  
Extension\_Selector, Model\_Type.

These models have a field that is a Field model :  
Model\_Type, Field\_Error.

The key field for this model is name  
This model may be extended with user-defined fields.

name -- a Symbol field of model Field  
The name of the Field, and thus the name of the function available in Expressions. This is only editable if 'user\_defined' is "true".  
Default: None -- this is a key field

type -- a Type field of model Field  
The Type of value returned by this Field. This is only editable if 'user\_defined' is "true".  
Default: String

description -- a String field of model Field  
A description of this Field.  
Default: none

default -- a Computed\_String field of model Field  
If a user defined field is not in a model, the field accessor returns NONEXISTENT. this field gives the ability to specify some other default.

Default: NONEXISTENT  
Properties: java\_method=default\_value

editable -- a Logical field of model Field  
TRUE if the field can be set, FALSE if it's read-only.  
Properties: Export-Only Field

field\_type -- an extension\_selector of model Field  
Defines the kind of field this is: SIMPLE, SELECTOR, EXTENDED, USER  
Default: SIMPLE  
Properties: Export-Only Field  
Extensions:  
SIMPLE, SELECTOR, EXTENDED, USER.

access -- a Expression field of model Field  
A logical expression that returns "true" if the current user can set this field The 'access' Expression is passed '#' (equal to this field's model). The current user is available via the 'user' worksheet function.  
Default: True

after\_set -- a Expression field of model Field  
An expression to execute after setting this field. The Expression in the is passed '#' (equal to this field's model). This can be useful for propagating changes in one field to other fields.  
Default: none

extension\_selector -- a Extension\_Selector field of model Field  
The field which can be set to one of the 'extensions' adding this Field to the model. If none, then this Field is always present in the model (it is not an extension Field).  
Properties: Export-Only Field

extensions -- a List(Symbol) field of model Field  
The extensions which add this Field to the model. If 'extension\_selector' is set to one of the Symbols in this List, then this field is added to the model.  
Properties: Export-Only Field

user\_defined -- a Logical field of model Field  
If "true", then this is a user-defined Field. Only user-defined Fields may be edited. If this returns "false", then none of the fields of this model may be edited.  
Properties: Export-Only Field

Connection Model	Field Model
------------------	-------------

value (Typed\_Value) - - *a Typed\_Value field of model Field*  
 Return the value for this field, given an instance of this field's model  
 Default: none  
 Properties: java\_method=field\_value

owner - - *a Model\_Type field of model Field*  
 The Model\_Type to which this Field belongs. The first argument to the Field function is a model of 'owner' Model\_Type.  
 Properties: Export-Only Field

Connection Model	Extension_Selector Model
------------------	--------------------------

### 7.2.2 Extension\_Selector Model

Extension\_Selector -- *a submodel of model Model\_Type*

An Extension\_Selector is a special field that can be set to one of the 'extensions', which adds fields to the model.

The Extension\_Selector model has fields that references these models :  
 Model\_Type.

These models have a field that is a Extension\_Selector model :  
 Model\_Type, Field.

The key field for this model is name

name - - *a Symbol field of model Extension\_Selector*  
 The name of the Field, and thus the name of the function available in Expressions. This is only editable if 'user\_defined' is "true".  
 Default: None -- this is a key field  
 Properties: Export-Only Field

description - - *a String field of model Extension\_Selector*  
 A description of this Field. This is only editable if 'user\_defined' is "true".  
 Default: none

extensions - - *a List(Symbol) field of model Extension\_Selector*  
 The extensions which can be selected.  
 Properties: Export-Only Field

selected\_fields (Symbol) - - *a List(Field) field of model Extension\_Selector*  
 All extension Fields added by a given extension value.  
 Properties: Export-Only Field

owner - - *a Model\_Type field of model Extension\_Selector*  
 The Model\_Type to which this Field belongs. The first argument to the Field function is a model of 'owner' Model\_Type.  
 Properties: Export-Only Field

Connection Model	Field_Error Model
------------------	-------------------

7.3 Field\_Error Model

Field\_Error -- an independent (top-level) model

When an error occurs while editing fields or importing data, it is recorded in this model. When the error is fixed, this model is automatically removed. In this way, erroneous data is never put into the model, but it is also not lost or forgotten. In lieu of erroneous data, default non-erroneous data will be substituted. The Field\_Error informs the user where that has been done so that the information can be fixed later.

The Field\_Error model has fields that references these models :  
Field.

The key field for this model is description

target\_model - - a Typed\_Value field of model Field\_Error

The model that had the error (the key-field of the target\_field's model). The Model\_Type of 'model' is the same as the 'owner\_model' of 'field'.  
Properties: Export-Only Field

target\_field - - a Field field of model Field\_Error  
The Field which had the error.  
Properties: Export-Only Field

error\_value - - a String field of model Field\_Error  
The erroneous value entered by user or read from data-file, based upon the Type of 'target\_field'. This is 'error\_input' converted to 'target\_field.type'.  
Properties: Export-Only Field

error\_input - - a String field of model Field\_Error  
The erroneous input String entered by a user or read from a data-file.  
Properties: Export-Only Field

description - - a String field of model Field\_Error  
A textual description of the error (what is wrong; what was done instead).  
Default: None -- this is a key field  
Properties: Export-Only Field

Connection Model	Field_Error Model
------------------	-------------------

source - - a String field of model Field\_Error  
The source of the error. If the error occurred while reading a file, this includes the file-name and line number. If the error occurred in an interactive Report, this includes name and cell location.  
Properties: Export-Only Field

Function Model	Field_Error Model
----------------	-------------------

### 8 Function Model

Function -- an independent (top-level) model

Breakpoint Model	Field_Error Model
------------------	-------------------

### 9 Breakpoint Model

Breakpoint -- an independent (top-level) model

# 10 Extensions

Site  
LINK  
SUPPLIER  
CUSTOMER  
Seller  
NONE  
Site\_Group  
Product\_Root  
SIMPLE\_FIXED\_QUANTITY  
SIMPLE\_REQUEST  
SIMPLE\_FIXED\_TIME  
WEEKLY  
DUAL\_REQUEST  
Product  
SIMPLE\_FIXED\_QUANTITY  
SIMPLE\_REQUEST  
SIMPLE\_FIXED\_TIME  
WEEKLY  
DUAL\_REQUEST  
AT\_END  
FCFS  
PER\_ALLOCATED  
PER\_COMMITTED  
MEMBER\_RANK  
FIXED\_SPLIT  
SLIDING  
HORIZON  
NONE  
FIXED  
Operation  
ALTERNATES\_PRIMARY  
ALTERNATES\_PROPORTIONAL  
EFFECTIVE\_CALENDAR  
DELAY\_ONLY\_FIXED

DELAY\_ONLY\_BASIC  
BASIC\_CALENDARS  
FIXED\_TIME  
TIME\_MULTIPLE  
BASIC  
BASIC\_DELAYED  
REQUEST\_FIXED  
REQUEST\_FIXED\_WITH\_ANALYSIS  
ROUTING  
Load  
RESOURCE  
ONE  
Load\_Size  
FIXED  
LINEAR  
Flow  
CONSUME\_FIXED  
CONSUME\_PER  
PRODUCE\_FIXED  
PRODUCE\_PER  
PRODUCE\_YIELD  
PRODUCE\_YIELD\_CALENDAR  
Operation\_Problem\_Detector  
Operation\_Plan  
PRODUCE  
CONSUME  
DELIVER  
RECEIVE  
ALTERNATES\_PRIMARY  
ALTERNATES\_PROPORTIONAL  
EFFECTIVE\_CALENDAR  
DELAY\_ONLY\_FIXED  
DELAY\_ONLY\_BASIC  
BASIC\_CALENDARS  
FIXED\_TIME  
TIME\_MULTIPLE

BASIC  
BASIC\_DELAYED  
REQUEST\_FIXED  
REQUEST\_FIXED\_WITH\_ANALYSIS  
ROUTING  
Resource  
FIXED  
CALENDAR  
ZERO  
FIXED  
ZERO  
UNLIMITED  
FIXED\_COUNT  
FIXED\_QUANTITY  
CALENDAR\_COUNT  
MULTI\_DIMENSION  
SIMPLE\_CONSOLIDATION  
INFINITE\_USE  
EXCLUSIVE\_USE  
SHARED\_USE  
Resource\_Skill  
FIXED  
CALENDAR  
Resource\_Plan  
SIMPLE\_CONSOLIDATION  
INFINITE\_USE  
EXCLUSIVE\_USE  
SHARED\_USE  
Buffer  
BUCKETED\_NESTED\_SORT  
PRODUCING\_FLOW\_CALENDAR  
PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK  
SUPPLY\_CALENDAR  
ON\_HAND\_CALENDAR  
ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK  
FLOW\_LIMIT\_CALENDAR

FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK  
INFINITE  
SUPPLIER  
BASIC  
BASIC\_FILTER\_AND\_RANK  
FIXED\_QUANTITY  
MULTIPLE  
MULTIPLE\_FILTER\_AND\_RANK  
FIXED\_QUANTITY\_FENCED  
FIXED\_TIME  
LFL\_SIMPLE  
LFL\_BOUNDED  
MLFL\_BOUNDED  
MANUAL  
CALENDAR  
Buffer\_Problem\_Detector  
Buffer\_Plan  
BUCKETED\_NESTED\_SORT  
PRODUCING\_FLOW\_CALENDAR  
PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK  
ON\_HAND\_CALENDAR  
ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK  
FLOW\_LIMIT\_CALENDAR  
FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK  
BASIC  
BASIC\_FILTER\_AND\_RANK  
FIXED\_QUANTITY  
MULTIPLE  
MULTIPLE\_FILTER\_AND\_RANK  
FIXED\_QUANTITY\_FENCED  
FIXED\_TIME  
Forecast  
INDIVIDUAL  
GROUP  
Forecast\_Entry  
MEMBER\_RANK

Site\_Plan

LINK

SUPPLIER

CUSTOMER

Problem

REQUEST\_NOT\_PLANNED  
REQUEST\_PLANNED\_LATE  
REQUEST\_PLANNED\_EARLY  
REQUEST\_PLANNED\_SHORT  
REQUEST\_PLANNED\_EXCESS  
PROMISE\_NOT\_PLANNED  
PROMISE\_PLANNED\_LATE  
PROMISE\_PLANNED\_EARLY  
PROMISE\_PLANNED\_SHORT  
PROMISE\_PLANNED\_EXCESS  
ACCEPTANCE\_NOT\_PLANNED  
ACCEPTANCE\_PLANNED\_LATE  
ACCEPTANCE\_PLANNED\_EARLY  
ACCEPTANCE\_PLANNED\_SHORT  
ACCEPTANCE\_PLANNED\_EXCESS  
PLANNED\_BEFORE\_CURRENT  
UNRELEASED  
NEEDS\_RELEASE  
INCONSISTENT\_OPPLAN  
REQUEST\_PROMISED\_LATE  
REQUEST\_PROMISED\_EARLY  
REQUEST\_PROMISED\_SHORT  
REQUEST\_PROMISED\_EXCESS  
ITEM\_PROMISE\_OVERPRICED  
DELIVERY\_PROMISE\_OVERPRICED  
PROMISE\_NOT\_OFFERED  
PROMISE\_NOT\_ACCEPTED  
ACCEPTANCE\_INCONSISTENT  
REQUEST\_QUEUED  
DELIVERY\_REQUEST\_NOT\_COORDINATED  
DELIVERY\_PROMISE\_NOT\_COORDINATED

DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED  
NEGATIVE\_ATP  
NEGATIVE\_PLANNED\_ATP  
OVER\_COMMITTED  
OVER\_CONSUMED  
UNALLOCATED\_FORECAST  
SUPPLY\_PLANNED\_LATE  
SUPPLY\_PLANNED\_EARLY  
SUPPLY\_PLANNED\_SHORT  
SUPPLY\_PLANNED\_EXCESS  
SUPPLY\_PROMISED\_LATE  
SUPPLY\_PROMISED\_EARLY  
SUPPLY\_PROMISED\_SHORT  
SUPPLY\_PROMISED\_EXCESS  
UNIDENTIFIED\_OP\_STATE  
UNCONSOLIDATED  
UNCOORDINATED  
CONSOLIDATION\_OVERSIZE  
CONSOLIDATION\_UNERSIZE  
OVERLOAD  
OVERSIZE  
BUCKET\_OVERSIZE  
UNDERLOAD  
NEGATIVE\_ON\_HAND  
OVER\_FLOW\_LIMIT  
NEGATIVE\_ON\_HAND\_AT\_END  
LOT\_OVER\_CONSUMED  
LOT\_NOT\_CONSUMED  
LOT\_NOT\_PRODUCED  
LOT\_OVER\_PRODUCED  
LOW\_ON\_HAND  
EXCESS\_ON\_HAND  
EXCESS\_ON\_HAND\_AT\_END  
Active\_Strategy  
NO\_PROBLEMS  
TARGET\_ACHIEVED

RESOLVE\_COUNT\_EXCEEDED  
MANUAL  
SEQUENCE\_RUN\_ONCE  
SEQUENCE\_RUN\_MULTIPLE  
BEFORE\_AND\_AFTER  
SEQUENTIAL\_ALTERNATES  
SEQUENTIAL\_ALTERNATES\_KEEP\_BEST  
PROPORTIONAL\_RESOLVES  
ORDERED\_RESOLVES  
PROPORTIONAL\_INTERACTION  
EARLIEST\_PROBLEM\_START  
SORT\_BY\_EXPRESSION

Active\_Goal

FEASIBILITY  
MINIMIZE\_PROBLEM\_COUNT  
MINIMIZE\_PROBLEMS  
MINIMIZE\_LATENESS  
WEIGHTED\_LATENESS  
WEIGHTED\_SHORTNESS  
MINIMIZE\_SHORTNESS  
MINIMIZE\_COST  
MAXIMIZE\_PROFIT  
MAXIMIZE\_REVENUE

Item

STANDARD  
CUSTOM

Skill

PRIMARY  
PREFER\_PRIMARY  
EVEN  
MAX\_EFFICIENCY  
Skill\_Resource  
FIXED  
CALENDAR  
Configuration  
STANDARD

CUSTOM  
Operation\_State  
EARLIEST  
STARTED  
COMPLETED  
IN\_FRONT

Request

FIXED  
NUMBERED

Delivery\_Request

BUCKETED\_ALL  
BUCKETED\_ASAP  
ON\_TIME

ALL

ALL\_ON\_TIME  
ASAP

ASAP\_MONTHLY

BUCKETED\_ALLOCATION  
BUCKETED\_ALL\_MIN\_PRICE  
BUCKETED\_MIN\_PRICE\_ASAP  
SHIP\_IN\_RATIO  
ON\_TIME

FULL\_QUANTITIES\_OF\_ALL\_ITEMS  
UNRESTRICTED

Promise

FIXED

Delivery\_Promise

ON\_TIME  
FULL\_QUANTITIES\_OF\_ALL\_ITEMS  
UNRESTRICTED

Horizon

ONE  
MONTHS  
WEEKS  
DAYS  
DATES



Delivery\_Acceptance  
ON\_TIME  
FULL\_QUANTITIES\_OF\_ALL\_ITEMS  
UNRESTRICTED

## Strategy

NO\_PROBLEMS  
TARGET\_ACHIEVED  
RESOLVE\_COUNT\_EXCEEDED  
MANUAL  
SEQUENCE\_RUN\_ONCE  
SEQUENCE\_RUN\_MULTIPLE  
BEFORE\_AND\_AFTER  
SEQUENTIAL\_ALTERNATES  
SEQUENTIAL\_ALTERNATES\_KEEP\_BEST  
PROPORTIONAL\_RESOLVES  
ORDERED\_RESOLVES  
PROPORTIONAL\_INTERACTION  
EARLIEST\_PROBLEM\_START  
SORT\_BY\_EXPRESSION

## Problem\_Set

REQUEST\_NOT\_PLANNED  
REQUEST\_PLANNED\_LATE  
REQUEST\_PLANNED\_EARLY  
REQUEST\_PLANNED\_SHORT  
REQUEST\_PLANNED\_EXCESS  
PROMISE\_NOT\_PLANNED  
PROMISE\_PLANNED\_LATE  
PROMISE\_PLANNED\_EARLY  
PROMISE\_PLANNED\_SHORT  
PROMISE\_PLANNED\_EXCESS  
ACCEPTANCE\_NOT\_PLANNED  
ACCEPTANCE\_PLANNED\_LATE  
ACCEPTANCE\_PLANNED\_EARLY  
ACCEPTANCE\_PLANNED\_SHORT  
ACCEPTANCE\_PLANNED\_EXCESS  
PLANNED\_BEFORE\_CURRENT

UNRELEASED  
NEEDS\_RELEASE  
INCONSISTENT\_OPPLAN  
OPERATION  
REQUEST\_PROMISED\_LATE  
REQUEST\_PROMISED\_EARLY  
REQUEST\_PROMISED\_SHORT  
REQUEST\_PROMISED\_EXCESS  
PROMISE\_NOT\_OFFERED  
PROMISE\_NOT\_ACCEPTED  
ACCEPTANCE\_INCONSISTENT  
REQUEST\_QUEUED  
REQUEST  
REQUEST\_PLAN  
PROMISE  
PROMISE\_PLAN  
REQUEST\_PROMISE  
DELIVERY\_REQUEST\_NOT\_COORDINATED  
DELIVERY\_PROMISE\_NOT\_COORDINATED  
DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED  
SUPPLY\_PLANNED\_LATE  
SUPPLY\_PLANNED\_EARLY  
SUPPLY\_PLANNED\_SHORT  
SUPPLY\_PLANNED\_EXCESS  
SUPPLY\_PROMISED\_LATE  
SUPPLY\_PROMISED\_EARLY  
SUPPLY\_PROMISED\_SHORT  
SUPPLY\_PROMISED\_EXCESS  
SUPPLY  
SUPPLY\_PLAN  
SUPPLY\_PROMISE  
UNIDENTIFIED\_OP\_STATE  
UNCONSOLIDATED  
UNCOORDINATED  
CONSOLIDATION\_OVERSIZE  
CONSOLIDATION\_UNDERSIZE

Extensions	Field_Error Model
------------	-------------------

OVERLOAD  
OVERSIZE  
BUCKET\_OVERSIZE  
UNDERLOAD  
RESOURCE  
NEGATIVE\_ON\_HAND  
OVER\_FLOW\_LIMIT  
NEGATIVE\_ON\_HAND\_AT\_END  
LOT\_OVER\_CONSUMED  
LOT\_NOT\_CONSUMED  
LOT\_NOT\_PRODUCED  
LOT\_OVER\_PRODUCED  
LOW\_ON\_HAND  
EXCESS\_ON\_HAND  
EXCESS\_ON\_HAND\_AT\_END  
BUFFER

Strategy\_Change  
MOVE\_IN  
MOVE\_OUT  
SPLIT  
USE\_MORE  
USE\_LESS  
USE\_ALTERNATE\_OPERATION  
USE\_ALTERNATE\_RESOURCE  
USE\_EFFECTIVE\_ALTERNATE  
INCREASE\_CAPACITY  
DECREASE\_CAPACITY  
RESIZE\_MORE  
RESIZE\_LESS  
UPSTREAM  
DOWNSTREAM  
Strategy\_Lock  
OPERATION\_PLAN\_RANK\_RANGE  
OPERATION\_PLAN\_RANK\_EXPRESSION  
Strategy\_Goal  
FEASIBILITY

Extensions	Field_Error Model
------------	-------------------

MINIMIZE\_PROBLEM\_COUNT  
MINIMIZE\_PROBLEMS  
MINIMIZE\_LATENESS  
WEIGHTED\_LATENESS  
MINIMIZE\_SHORTNESS  
WEIGHTED\_SHORTNESS  
MINIMIZE\_COST  
MAXIMIZE\_PROFIT  
MAXIMIZE\_REVENUE  
Calendar\_Entry  
NUMBER  
QUANTITY  
NUMBER\_QUANTITY  
SYMBOL  
TIME  
EVERYDAY  
EVERY\_N\_DAYS  
WEEKDAYS  
WEEKENDS  
DAYS\_OF\_WEEK  
DAY\_OF\_MONTH  
DAY\_OF\_WEEK\_OF\_MONTH  
DAY\_OF\_LAST\_WEEK\_OF\_MONTH  
DAY\_OF\_YEAR  
YEARLY  
Calendar  
UNSPECIFIED  
NUMBER  
QUANTITY  
NUMBER\_QUANTITY  
SYMBOL  
TIME  
Flow\_Criterion  
CUSTOMER\_RANK  
SELLER\_RANK  
REQUEST\_RANK

ACTUAL\_OR\_FORECAST  
REQUEST\_ISSUED  
PROMISE\_DUE  
REQUEST\_DUE  
DUE\_DATE  
PROMISED  
ENTRY\_DATE  
Calendar\_Plan  
NUMBER  
QUANTITY  
NUMBER\_QUANTITY  
SYMBOL  
TIME  
Format  
Void  
Logical  
String  
Symbol  
Date  
Date\_Range  
Number  
Percentage  
Integer  
Quantity  
Quantity\_Range  
Time  
Restriction  
Horizon\_Date  
List  
Field  
SIMPLE  
SELECTOR  
EXTENDED  
USER

10.1 Site Extensions

10.1.1 role extensions of model Site

LINK -- a role extension of model Site

A LINK Site is a "link" in this supply "chain". It purchases Items from zero or more other Sites, performs Operations on Items, moving or transforming them, and supplies Items to zero or more other Sites.

LINK Sites can be modeled in detail. They consist of Operations which use Resources to transform or transfer Items between Buffers (the FLO network). They place Requests on other LINK and SUPPLIER Sites, and provide Promises to other LINK and CUSTOMER Sites.

The LINK model has these submodels :  
Location, Item, Buffer, Resource, Skill, Operation, Configuration, Item\_Group.

The LINK model has fields that references these models :  
Location, Item, Buffer, Resource, Skill, Operation, Configuration, Item\_Group.

managed - - - a Logical field of model LINK  
If "true", then this LINK Site is planned and managed by this model. The plans created for such a Site are assumed to be the plans it will follow.

If "false", then this LINK Site is managed elsewhere and this model is simply modeling its behavior. Such modeling is purely informational -- the plans created for such a Site cannot be assumed to be reliable. Only the Promises provided from the managers of that Site can be relied upon (hopefully, anyway).

Requests and Promises will be created between all unmanaged Sites and the managed Sites, just as would be created with SUPPLIER Sites and CUSTOMER Sites. In contrast, managed Sites will interface directly to each other. There is no sense in a planner making Requests to himself or responding with formal Promises to himself. Requests and Promises will be created for informational and reporting purposes, but they will be maintained automatically -- as if there was no Site boundary at all.

locations - - - a list of Location submodels of model LINK  
The Locations within this Site.

Extensions	LINK Extension
------------	----------------

**top\_locations** -- *a List[Location] field of model LINK*  
 The Locations within this Site whose 'super\_location' is [unspecified]. These are the topmost Locations at this Site. This is a useful starting point for generating a Location hierarchy.  
 Properties:   Export-Only Field

**items** -- *a list of Item submodels of model LINK*  
 The Items consumed, used, and/or produced at this Site. Requests placed on this Site must be for Items in this list that are 'sellable'.  
 Properties:   Export-Only Field

**top\_items** -- *a List[Item] field of model LINK*  
 The Items within this Site whose 'family' is [unspecified]. These are the topmost Locations at this Site. This is a useful starting point for generating an Item hierarchy.  
 Properties:   Export-Only Field

**buffers** -- *a list of Buffer submodels of model LINK*  
 The Buffers to be used for managing the flow of Items at this Site.

**resources** -- *a list of Resource submodels of model LINK*  
 The Resources which model the capacity to perform Operations at this Site.

**skills** -- *a list of Skill submodels of model LINK*  
 The Skills needed to perform Operations; Resources have these Skills.

**operations** -- *a list of Operation submodels of model LINK*  
 The Operations that transform Items into other Items and/or move Items between Buffers.

**configurations** -- *a list of Configuration submodels of model LINK*  
 Should be hidden. The list of these is not relevant.

**item\_groups** -- *a list of Item\_Group submodels of model LINK*  
 Item\_Groups group items into hierarchies. Each item can appear in at most one Item\_Group in a hierarchy of Item\_Groups. But each item can appear in any number of independent Item\_Group hierarchies.

**top\_item\_groups** -- *a List[Item\_Group] field of model LINK*  
 This List is a subset of item\_groups consisting only of Item\_Groups that are the top of a Item\_Group hierarchy, the Item\_Groups that do not appear within any other Item\_Group.  
 Properties:   Export-Only Field

Extensions	SUPPLIER Extension
------------	--------------------

**buffers\_at\_level (Integer)** -- *a List[Buffer] field of model LINK*  
 A List of the Buffers at a particular Integer 'level'. Note, 'site.buffers\_at\_level(3)' is equivalent to 'site.buffers.filter{#.level == 3}', but is likely less expensive to compute.  
 Properties:   Export-Only Field

**operations\_at\_level (Integer)** -- *a List[Operation] field of model LINK*  
 A List of the Operations at a particular Integer 'level'. An operation level is defined as the min buffer level of all its consuming flow buffers.  
 Properties:   Export-Only Field

**buffer\_levels** -- *a List[Integer] field of model LINK*  
 A List of all the level Integer values used by any of 'buffers'. It is a List containing 0, and the each Integer in order up to the largest 'level' code of any of 'buffers'. This List is generally used to feed the 'buffers\_at\_level' function in order to generate a table of the Buffers by-level.  
 Properties:   Export-Only Field

**operation\_levels** -- *a List[Integer] field of model LINK*  
 A List of all the level Integer values used by any of 'operations'. This List is generally used to feed the 'operations\_at\_level' function in order to generate a table of the Operations by-level.  
 Properties:   Export-Only Field

#### SUPPLIER -- a role extension of model Site

A SUPPLIER Site is not planned or modeled in detail; rather it represents the Items that can be procured from a supplier by LINK Sites, the Requests for those Items, and the Promises received from the supplier.

The SUPPLIER model has these submodels :

Item.

The SUPPLIER model has fields that references these models :

Item.

**items** -- *a list of Item submodels of model SUPPLIER*  
 The Items produced at this Site. Requests placed on this Site must be for one of these Items.

#### CUSTOMER -- a role extension of model Site

<i>Extensions</i>	<i>CUSTOMER Extension</i>
-------------------	---------------------------

A CUSTOMER Site is not planned or modeled in detail; rather, it is simply a destination for deliveries and source of Requests.

<i>Extensions</i>	<i>Seller Extensions</i>
-------------------	--------------------------

10.2 Seller Extensions

10.2.1 request\_naming extensions of model Seller

NONE -- a request\_naming extension of model Seller

Seller's Requests are not named.

Extensions	Site_Group Extensions
------------	-----------------------

### 10.3 Site\_Group Extensions

#### 10.3.1 spec extensions of model Site\_Group

Extensions	Product_Root Extensions
------------	-------------------------

### 10.4 Product\_Root Extensions

#### 10.4.1 forecast\_policy extensions of model Product\_Root

##### SIMPLE\_FIXED\_QUANTITY -- a forecast\_policy extension of model Product\_Root

The Product is forecasted by a single quantity, the total delivered during the forecast Date\_Range. The 'order\_quantity' defines the expected order size. All generated forecast Requests, except possibly the final one, will have the same Quantity which will be spread evenly through the dates covered by that Forecast\_Entry. The final request may be less than 'order\_quantity' as the total number requested will be equal to the committed forecast.

No compensation is made for forecast error, nor for variance in timing. As current time advances and passes the generated forecast Requests, those forecasts are expired and eliminated.

The SIMPLE\_FIXED\_QUANTITY model has fields that references these models : Configuration.

order\_quantity -- a Quantity field of model SIMPLE\_FIXED\_QUANTITY

The high end of the common requested Quantity's of this Product.

Default: 00

representative\_configuration -- a Configuration field of model SIMPLE\_FIXED\_QUANTITY

The Configuration used to represent the production of this Product during master planning, when planning to forecast prior to knowing exactly what configurations will be requested.

Note that this forecast\_policy creates forecast\_requests based on a representative item, which must be specified in the 'representative\_configuration' in order for forecast\_requests to be generated properly. Failure to specify the 'representative\_configuration' item will result in forecast\_requests for the [unspecified] item of the [unspecified] site. (The only exception to this rule is the case in which the Product\_Root has only one Product\_Item. In this case, it is assumed that the single Product\_Item is intended to be the 'representative\_configuration.item' 100.)

Properties: Export-Only Field

representative\_quantity\_per -- a Quantity field of model SIMPLE\_FIXED\_QUANTITY

The Quantity of 'representative\_configuration' planned per unit of this Product.

Default: 1

rough\_fence - - a Horizon\_Date field of model SIMPLE\_FIXED\_QUANTITY

At and after the 'rough\_fence', this forecast policy behaves as a SINGLE\_REQUEST forecast policy.

Default: infinite future

SINGLE\_REQUEST -- a forecast\_policy extension of model Product\_Root

The Product is forecasted by a single quantity, the total delivered during the forecast Date\_Range. One forecast Request will be generated at the beginning of each forecast\_period, having the same Quantity as 'committed' for that Forecast\_Entry.

No compensation is made for forecast error, nor for variance in timing. As current time advances and passes the generated forecast Requests, those forecasts are expired and eliminated.

The SINGLE\_REQUEST model has fields that references these models : Configuration.

representative\_configuration - - a Configuration field of model SINGLE\_REQUEST

The Configuration used to represent the production of this Product during master planning, when planning to forecast prior to knowing exactly what configurations will be requested.

Note that this forecast\_policy creates forecast\_requests based on a representative item, which must be specified in the 'representative\_configuration' in order for forecast\_requests to be generated properly. Failure to specify the 'representative\_configuration's item will result in forecast\_requests for the [unspecified] item of the [unspecified] site. (The only exception to this rule is the case in which the Product\_Root has only one Product\_Item. In this case, it is assumed that the single Product\_Item is intended to be the 'representative\_configuration.item' too.) Properties: Export-Only Field

representative\_quantity\_per - - a Quantity field of model SINGLE\_REQUEST  
The Quantity of 'representative\_configuration' planned per unit of this Product.

Default: 1

SIMPLE\_FIXED\_TIME -- a forecast\_policy extension of model Product\_Root

The Product is forecasted by a single quantity, the total delivered during the forecast Date\_Range. The generated forecast requests will be placed at the beginning of the bucket and then at each 'interval' of time through the rest of the bucket. The requests will be for approximately the same quantity, though may be rounded up or down to the Item's 'unit'. The sum of the requests will be the forecasted quantity rounded up to the Item's 'unit'.

No compensation is made for forecast error, nor for variance in timing. As current time advances and passes the generated forecast Requests, those forecasts are expired and eliminated.

The SIMPLE\_FIXED\_TIME model has fields that references these models : Configuration.

interval - - a Time field of model SIMPLE\_FIXED\_TIME

The Time between the generated forecast Delivery\_Requests.

Default: 1 week

representative\_configuration - - a Configuration field of model SIMPLE\_FIXED\_TIME

The Configuration used to represent the production of this Product during master planning, when planning to forecast prior to knowing exactly what configurations will be requested.

Note that this forecast\_policy creates forecast\_requests based on a representative item, which must be specified in the 'representative\_configuration' in order for forecast\_requests to be generated properly. Failure to specify the 'representative\_configuration's item will result in forecast\_requests for the [unspecified] item of the [unspecified] site. (The only exception to this rule is the case in which the Product\_Root has only one Product\_Item. In this case, it is assumed that the single Product\_Item is intended to be the 'representative\_configuration.item' too.) Properties: Export-Only Field

representative\_quantity\_per - - a Quantity field of model SIMPLE\_FIXED\_TIME

The Quantity of 'representative\_configuration' planned per unit of this Product.

Default: 1

rough\_fence - - a Horizon\_Date field of model SIMPLE\_FIXED\_TIME  
At and after the 'rough\_fence', this forecast policy behaves as a SINGLE\_REQUEST forecast policy.

Default: infinite future

<i>Extensions</i>	<i>WEEKLY Extension</i>
-------------------	-------------------------

**WEEKLY -- a forecast\_policy extension of model Product\_Root**

The Product is forecasted by a single quantity, the total delivered during the forecast Date\_Range. The 'day\_of\_week' defines the expected delivery day. The generated forecast request will be placed on day\_of\_week at one week intervals of time throughout the bucket. The request will be for approximately the same quantity, though they may be rounded up or down to the Item's unit'. The sum of the requests will be the forecasted quantity rounded up to the Item's unit'.

No compensation is made for forecast error, nor for variance in timing. As current time advances and passes the generated forecast Requests, those forecasts are expired and eliminated.

The WEEKLY model has fields that references these models :  
Configuration.

day\_of\_week - - a Integer field of model WEEKLY

The day of each week that the Delivery\_Request should be due. Monday is day 1; Sunday is day 7.

Default: Monday

representative\_configuration - - a Configuration field of model WEEKLY

The Configuration used to represent the production of this Product during master planning, when planning to forecast prior to knowing exactly what configurations will be requested.

Note that this forecast\_policy creates forecast\_requests based on a representative item, which must be specified in the 'representative\_configuration' in order for forecast\_requests to be generated properly. Failure to specify the representative\_configuration's item will result in forecast\_requests for the [unspecified] item of the [unspecified] site. (The only exception to this rule is the case in which the Product\_Root has only one Product\_Item. In this case, it is assumed that the single Product\_Item is intended to be the 'representative\_configuration.item' (100.)

Properties: Export-Only Field

representative\_quantity\_per - - a Quantity field of model WEEKLY

The Quantity of 'representative\_configuration' planned per unit of this Product.

Default: 1

<i>Extensions</i>	<i>DUAL_REQUEST Extension</i>
-------------------	-------------------------------

rough\_fence - - a Horizon\_Date field of model WEEKLY

At and after the 'rough\_fence', this forecast policy behaves as a SINGLE\_REQUEST forecast policy.

Default: infinite future

**DUAL\_REQUEST -- a forecast\_policy extension of model Product\_Root**

The Product forecast demand is placed on the representative item's site via dual forecast Requests.

One forecast Request represents the amount of the Forecast\_Entry's committed quantity which has not been covered by any allocation (i.e. forecast\_entry.committed - forecast\_entry.allocated). This forecast Request represents the amount "still needed" to satisfy the forecast.

The other forecast Request represents the amount of the Forecast\_Entry's committed quantity which has been covered by an allocation but has not yet been consumed (i.e. forecast\_entry.allocated - forecast\_entry.consumed). This forecast Request represents the "unconsumed allocation" of the forecast\_entry.

No compensation is made for forecast error, nor for variance in timing. As current time advances and passes the generated forecast Requests, those forecasts are expired and eliminated.

The DUAL\_REQUEST model has fields that references these models :  
Configuration.

representative\_configuration - - a Configuration field of model DUAL\_REQUEST

The Configuration used to represent the production of this Product during master planning, when planning to forecast prior to knowing exactly what configurations will be requested.

Note that this forecast\_policy creates forecast\_requests based on a representative item, which must be specified in the 'representative\_configuration' in order for forecast\_requests to be generated properly. Failure to specify the representative\_configuration's item will result in forecast\_requests for the [unspecified] item of the [unspecified] site. (The only exception to this rule is the case in which the Product\_Root has only one Product\_Item. In this case, it is assumed that the dual Product\_Item is intended to be the 'representative\_configuration.item' (100.)

Properties: Export-Only Field



**representative\_quantity\_per** -- *a Quantity field of model DUAL\_REQUEST*  
The Quantity of 'representative\_configuration' planned per unit of this Product.  
Default: 1

**rough\_fence** -- *a Horizon\_Date field of model DUAL\_REQUEST*  
At and after the 'rough\_fence', this forecast policy behaves as a SINGLE\_REQUEST forecast policy.  
Default: infinite future

10.5 Product Extensions

10.5.1 forecast\_policy extensions of model Product

**SIMPLE\_FIXED\_QUANTITY** -- *a forecast\_policy extension of model Product*

A **SIMPLE\_FIXED\_QUANTITY** Product is defined by a **SIMPLE\_FIXED\_QUANTITY** Product\_Root. See the documentation there.

**SINGLE\_REQUEST** -- *a forecast\_policy extension of model Product*

A **SINGLE\_REQUEST** Product is defined by a **SINGLE\_REQUEST** Product\_Root. See the documentation there.

**SIMPLE\_FIXED\_TIME** -- *a forecast\_policy extension of model Product*

A **SIMPLE\_FIXED\_TIME** Product is defined by a **SIMPLE\_FIXED\_TIME** Product\_Root. See the documentation there.

**WEEKLY** -- *a forecast\_policy extension of model Product*

A **WEEKLY** Product is defined by a **WEEKLY** Product\_Root. See the documentation there.

**DUAL\_REQUEST** -- *a forecast\_policy extension of model Product*

A **DUAL\_REQUEST** Product is defined by a **DUAL\_REQUEST** Product\_Root. See the documentation there.

10.5.2 expiration\_policy extensions of model Product

**AT\_END** -- *a expiration\_policy extension of model Product*

The forecast does not expire until the end of the forecast period. So, if 1000 units are forecasted, but only 10 units have actually been requested through the 26th day of the month period, the other 990 units will continue to be forecasted to be sold in the last few days.

10.5.3 allocation\_policy extensions of model Product

**FCFS** -- *a allocation\_policy extension of model Product*

The **FCFS** allocation\_policy does no allocation. Rather it makes the whole allocation available at this Seller for the 'members' to consume first-come first-served (FCFS).

PER\_ALLOCATED -- a allocation\_policy extension of model Product

The PER\_ALLOCATED allocation\_policy distributes initial allocations proportional to the quantity 'committed' to by that Seller. Thereafter, allocations are adjusted proportionally to the quantity 'allocated' to that Seller. A portion of the allocation can be retained for use at the discretion of this Seller. Any leftovers due to lot sizing or explicit adjustments will be available on a first-come-first-served (FCFS) basis.

retain -- a Percentage field of model PER\_ALLOCATED

The Percentage of the allocated Quantity that should be retained for use at the discretion of this Seller. No 'member' will get access to it unless the Seller explicitly reduces the 'retained' Quantity in a bucket.

Default: 0%

PER\_COMMITTED -- a allocation\_policy extension of model Product

The PER\_COMMITTED allocation\_policy distributes the allocations proportional to the quantity 'committed' to by that Seller. A portion of the allocation can be retained for use at the discretion of this Seller. Any leftovers due to lot sizing or explicit adjustments will be available on a first-come-first-served (FCFS) basis.

retain -- a Percentage field of model PER\_COMMITTED

The Percentage of the allocated Quantity that should be retained for use at the discretion of this Seller. No 'member' will get access to it unless the Seller explicitly reduces the 'retained' Quantity in a bucket.

Default: 0%

MEMBER\_RANK -- a allocation\_policy extension of model Product

The MEMBER\_RANK allocation\_policy distributes the allocations to the 'members' in the order of their 'rank'. For Sellers with equal 'rank', it distributes to each proportionally to the quantity 'committed' to by that Seller. A portion of the allocation can be retained for use at the discretion of this Seller. Any leftovers due to lot sizing or explicit adjustments will be available on a first-come-first-served (FCFS) basis. For example: Let us take a hypothetical situation, in which we have the following Seller hierarchy with member ranks shown in parentheses. Their forecasts are shown next to their ranks. Initial allocations for all sellers is zero. After planning, the promise is, say 2000. Now the Top Seller needs to distribute the change in the allocation (2000-0 = 2000) to its members. When there is an increase in allocation at a seller, we take the member with the highest rank. In this case we have two members with equal ranks. Now distribute 2000 between Northern Sales and southern Sales proportional to their committed (1500:1000 = 3:2). So the Northern Sales gets 1200 and the Southern Sales

gets 800. Eastern Sales does not get anything. Now we need to distribute 1200 (1200-0) between N1 Sales and N2 Sales. When there is an increase in allocation to the Northern Sales, we take the highest ranked member, i.e N2 Sales, and allocate 250 to him. Then we assign 250 for N1 Sales. Northern Sales keeps the rest, i.e. 700 (1200 - (250+250)).

Now the promise was changed to, say 800. Now we need to allocate -1200 (800-2000) to the members of the Top Seller. When there is a reduction in allocation, we take the lowest ranked member and try to make his net allocation zero. So we take Eastern Sales. His allocation is already zero. So we can't give (take from) him any more. Now take the next lowest ranked seller, Northern Sales and Southern Sales. Now try to allocate -1200 to these sellers proportional to their committed. Northern Sales gets -720 and Southern Sales gets -480. So their net allocations becomes -480 (1200+(-720)) and 320 (800+(-480)). Now the Northern Sales need to distribute -720 between him self and his sub sellers. We try to make the allocation to the Northern Sales' forecast (1500 - (250+250) = 1000) to zero. So we assign -700 to him. So we have -20 to distribute between N1 and N2. We take the lowest ranked member and try to assign -20. So we assign -20 to N1 Sales. His net allocation becomes 230. Allocations for N2 Sales does not change.

					Top Seller
(10)	3000				
1		1		1	Northern Sales (5) 1500
500	Southern Sales (5) 1000		1		Eastern Sales (1)
(2)	250	N2 Sales (3) 250		1	N1 Sales

retain -- a Percentage field of model MEMBER\_RANK

The Percentage of the allocated Quantity that should be retained for use at the discretion of this Seller. No 'member' will get access to it unless the Seller explicitly reduces the 'retained' Quantity in a bucket.

Default: 0%

pool\_allocation -- a Logical field of model MEMBER\_RANK

If this flag is true then allocation for the member sellers whose 'lock\_allocated' flag is true will be pooled at the organization level. This quantity will not be available to the organization or other members of the organization. It will be reserved for the members whose 'lock\_allocated' is true.

Default: FALSE

reallocate (Plan) -- a Void field of model MEMBER\_RANK

Run the allocation policy on the forecast in the specified plan ignoring the previous allocations. The 'allocated' field of member seller is populated in the following order. First try to meet the consumed quantity. Then 'accepted' quantity and finally 'committed' quantity  
Properties:    command=True    Export-Only Field

**FIXED\_SPLIT -- a allocation\_policy extension of model Product**

The FIXED\_SPLIT allocation\_policy distributes according a fixed percentage breakdown among the members of this Product's owner. This breakdown is defined in the 'allocations' sub-model. All members of the Product's owner which are not assigned a fixed percentage split in the 'allocations' list are assumed to be allocated zero.

A portion of the allocation can be 'retain'd for use at the discretion of this Seller. Any leftovers due to lot sizing or explicit adjustments will be available on a first-come-first-served (FCFS) basis.

If the sum of the fixed percentage splits defined in 'allocations' is not equal to 100%, then the 'splits' are normalized to 100%. Then the original amount is reduced by the 'retain' percent and the result is allocated using the normalized 'splits'.

The FIXED\_SPLIT model has these submodels :  
Product\_Allocation.

The FIXED\_SPLIT model has fields that references these models :  
Product\_Allocation.

retain -- a Percentage field of model FIXED\_SPLIT

The Percentage of the allocated Quantity that should be retained for use at the discretion of this Seller. No 'member' will get access to it unless the Seller explicitly reduces the 'retained' Quantity in a bucket.

Default:    0%

allocations -- a list of Product\_Allocation submodels of model FIXED\_SPLIT  
The 'allocations' or fixed percentages which are used by this product's owner when splitting its allocated amount up among its members. Any entry of 'allocations' whose split is set to zero is automatically deleted.

**10.5.4    availability\_policy extensions of model Product**

**SLIDING -- a availability\_policy extension of model Product**

The SLIDING availability policy represents the default behavior of the owner Product's 'consume\_earlier' field. This behavior allows modelling of ATP which expires over time, by establishing a sliding window of allocation availability. The width of the sliding window is defined by the 'consume\_earlier' field.

Unless otherwise stated, this SLIDING behavior is baseline functionality which is included in all other 'availability\_policies'.

**HORIZON -- a availability\_policy extension of model Product**

The HORIZON availability policy represents the addition of a series of fixed boundaries to the baseline SLIDING functionality. The fixed boundaries are specified by setting the horizon's bucket\_spec to the appropriate extension. This behavior allows modeling of ATP which both expires gradually over time (the baseline functionality defined by 'consume\_earlier') and also expires completely at a fiscal boundary (such as month-end or quarter-end).

The HORIZON model has fields that references these models :  
Horizon.

availability\_horizon -- a Horizon field of model HORIZON

Define how the Product's 'availability\_dates' are computed. This Horizon defines the availability boundaries which will limit the accumulation of ATP for this Product.  
Default:    [unspecified]

buckets (Date\_Range) -- a List(Date\_Range) field of model HORIZON

This will return list of buckets defined by currently selected 'bucket\_spec' extension.

If no argument is passed, this returns list of buckets within planning horizon(plan.start to plan.end). Start date for the first bucket will be same as the plan.start and the end date for the last bucket in the list will be same as plan.end.    !!!UNIMPLEMENTED!!!!

If Date\_Range argument is given, this will return list of buckets within that Date\_Range. First bucket's start will be same as the start date of the Date\_Range passed in as argument. Similarly, the end date of last bucket will be same as the end date of the Date\_Range passed in as argument.  
Properties:    Export-Only Field

Extensions	price_policy extensions of model Product
------------	--

10.5.5 price\_policy extensions of model Product  
NONE -- a price\_policy extension of model Product

The price\_policy, NONE, is used when a Product's pricing is not to be considered. All ATP prices for such a Product will be 0.

FIXED -- a price\_policy extension of model Product

The FIXED price\_policy supports fixed pricing. The price for a Product may be specified in the price' field. All ATP prices for that Product will then reflect the specified price.

price -- a Money field of model FIXED  
The base price for 1 unit of this Product.  
Default: 0

Extensions	Operation Extensions
------------	----------------------

10.6 Operation Extensions

10.6.1 process extensions of model Operation

ALTERNATES\_PRIMARY -- a process extension of model Operation

An ALTERNATES\_PRIMARY Operation executes one of the sub\_operations. It defaults to using the primary Operation, the one with the largest 'percentage'. When it needs to offload, it selects an alternate probabilistically, selecting proportional to 'percentage'.

The difference between ALTERNATES\_PRIMARY and ALTERNATES\_PROPORTIONAL is that ALTERNATES\_PRIMARY always selects the one with the largest 'percentage' initially, but ALTERNATES\_PROPORTIONAL makes the initial selection probabilistically also.

The ALTERNATES\_PRIMARY model has these submodels :  
Alternate\_Operation.

The ALTERNATES\_PRIMARY model has fields that references these models :  
Alternate\_Operation.

splittable -- a Logical field of model ALTERNATES\_PRIMARY  
This field exists only for backward compatibility. It will be implemented in a future release.  
Default: false  
Properties: obsolete=True

alternates -- a list of Alternate\_Operation submodels of model ALTERNATES\_PRIMARY  
The alternate Operations that may be selected to perform this Operation. Each of these alternates will show up in 'sub\_operations' of this Operation, but only the selected alternate will show up in 'sub\_operation\_plans' of an Operation\_Plan of this Operation.

ALTERNATES\_PROPORTIONAL -- a process extension of model Operation

An ALTERNATES\_PROPORTIONAL Operation executes one of the sub\_operations. It chooses which alternate to use probabilistically, weighted by the Alternate\_Operation's 'percentage'.

The ALTERNATES\_PROPORTIONAL model has these submodels :

Extensions	EFFECTIVE_CALENDAR Extension
------------	------------------------------

Alternate\_Operation.

The ALTERNATES\_PROPORTIONAL model has fields that references these models  
Alternate\_Operation.

splittable - - a Logical field of model ALTERNATES\_PROPORTIONAL  
This field exists only for backward compability. It will be implemented in a future release.

Default: false  
Properties: obsolete=True

alternates - - a list of Alternate\_Operation submodels of model ALTERNATES\_PROPORTIONAL

The alternate Operations that may be selected to perform this Operation. Each of these alternates will show up in 'sub\_operations' of this Operation, but only the selected alternate will show up in 'sub\_operation\_plans' of an Operation\_Plan of this Operation.

EFFECTIVE\_CALENDAR -- a process extension of model Operation

An EFFECTIVE\_CALENDAR Operation executes one of the sub\_operations. Selection of the sub operation will be done based on the effectivity specified by the calendar for that sub operation. Each sub operation can specify their own effectivity calendar. If more then one sub operation is effective on a given date, probabilistic selection will be done based on the 'percentage' value.

The EFFECTIVE\_CALENDAR model has these submodels :  
Effective\_Calendar\_Operation.

The EFFECTIVE\_CALENDAR model has fields that references these models :  
Effective\_Calendar\_Operation.

splittable - - a Logical field of model EFFECTIVE\_CALENDAR  
This field exists only for backward compability. It will be implemented in a future release.  
Default: false  
Properties: obsolete=True

Extensions	DELAY_ONLY_FIXED Extension
------------	----------------------------

effective\_alternates - - a list of Effective\_Calendar\_Operation submodels of model EFFECTIVE\_CALENDAR

The alternate Operations that may be selected to perform this Operation. Each of these alternates will show up in 'sub\_operations' of this Operation, but only the selected alternate will show up in 'sub\_operation\_plans' of an Operation\_Plan of this Operation.

DELAY\_ONLY\_FIXED -- a process extension of model Operation

A DELAY\_ONLY\_FIXED Operation loads no Resource. It simply runs for a fixed amount of time, independent of the quantity being processed. Thus, this can be used to insert a fixed delay in the processing. It is also commonly used to model fixed-lead-time Operations, which are particularly common when modeling outsourcing and purchased items.

time - - a Time field of model DELAY\_ONLY\_FIXED  
The fixed amount of Time that this Operation runs (delays).  
Default: 0

DELAY\_ONLY\_BASIC -- a process extension of model Operation

A DELAY\_ONLY\_BASIC Operation loads no Resource. It simply runs for a time equal to 'fixed\_time + units\*time\_per'. Thus, this can be used to insert a per-unit delay in the processing.

time - - a Time field of model DELAY\_ONLY\_BASIC  
The fixed additional Time that this Operation runs (delays), beyond the computed per-unit Time.  
Default: 0  
time\_per - - a Time field of model DELAY\_ONLY\_BASIC  
The Time per unit of the Operation that this Operation runs (delays).  
Default: 0

BASIC\_CALENDARS -- a process extension of model Operation

A BASIC\_CALENDARS Operation loads the Resources for a time equal to time + units\*time\_per/resource.efficiency', where both the time per unit of the operation (time\_per) and the additional operation time (time) are specified using calendars. In calculating the overall operation time, the amount of 'fixed time' (using time\_calendar) is scheduled before the 'per-unit time' chronologically in the operation plan. The algorithm used will produce the same results whether the operation is planned end-backward or start-forward. Both the 'fixed time' component and the per-

Extensions	FIXED_TIME Extension
------------	----------------------

unit time' component are calculated using a weighted average of the calendar entries specified for those times, over the required time ranges. For example, assume an operation plan's 'fixed time' segment overlaps two calendar entries where the latter entry specifies 5 days of fixed time, and the former entry specifies 10 days of fixed time. If the operation plan overlaps the 5-day segment by three days, this comprises 60 percent of the required fixed time (3 days where 5 are required). This means that 40 percent of the required fixed time (of 100 percent) must come from the segment where 10 days of fixed time are required. So when 10 days are required, 40 percent of that is 4 days. Therefore, the total fixed time for this operation will be 7 days, using the weighted average of the calendar entries. The actual number of calendar entries used in computing fixed time and per-unit time may be greater than or less than two, depending on the particular characteristics of the plan being generated.

The BASIC\_CALENDARS model has fields that references these models :  
Calendar.

**time\_calendar** - - *a Calendar field of model BASIC\_CALENDARS*  
The additional standard (100% efficiency) Time that this Operation runs beyond the computed per-unit Time. This time will vary, depending on the specified calendar. The time used will be taken from the calendar on the day the operation is scheduled to begin. The time specified will be multiplied by the loaded Resources efficiencies to determine actual time.  
Default: [unspecified]

**time\_per\_calendar** - - *a Calendar field of model BASIC\_CALENDARS*  
The standard (100% efficiency) Time per unit of the Operation that this Operation runs, as specified in a calendar. The time specified will be multiplied by the loaded Resources efficiencies to determine actual time.  
Default: [unspecified]

#### FIXED\_TIME -- a process extension of model Operation

A FIXED\_TIME Operation loads a Resource for a fixed amount of Time, independent of the Quantity being processed. It is the same as BASIC with time\_per equal to zero (except that this is smaller and faster).

**time** - - *a Time field of model FIXED\_TIME*  
The fixed time that this Operation runs. Note that this is standard time which will be multiplied by the loaded Resources' efficiencies to determine actual time.  
Default: 0

Extensions	TIME_MULTIPLE Extension
------------	-------------------------

#### TIME\_MULTIPLE -- a process extension of model Operation

A TIME\_MULTIPLE Operation loads a Resource for a Time that increases in discrete multiples of base\_time', based on the number of units as a multiple of base\_units. More mathematically, the time is round\_up(units / base\_units) \* base\_time'.

**base\_time** - - *a Time field of model TIME\_MULTIPLE*  
The base Time per base\_unit' that this Operation runs. Note that this is standard time which will be multiplied by the loaded Resources' efficiencies to determine actual time.  
Default: 0

**base\_units** - - *a Number field of model TIME\_MULTIPLE*  
The base number of units that can be run in base\_time'.  
Default: 1

#### BASIC -- a process extension of model Operation

A BASIC Operation loads the Resources for a time equal to time + units\*time\_per/resource\_efficiency'.

**time** - - *a Time field of model BASIC*  
The fixed additional Time that this Operation runs, beyond the computed time\_per' unit. Note that this is standard time which will be multiplied by the loaded Resources' efficiencies to determine actual time.  
Default: 0

**time\_per** - - *a Time field of model BASIC*  
The standard (100% efficiency) Time per unit of this Operation that this Operation runs. Note that this is standard time which will be multiplied by the loaded Resources' efficiencies to determine actual time.  
Default: 0

#### BASIC\_DELAYED -- a process extension of model Operation

A BASIC\_DELAYED Operation loads the Resources for a time equal to time + units\*time\_per/resource\_efficiency'. In addition, the BASIC\_DELAYED Operation total time includes a fixed 'warm\_up\_delay' that occurs prior to loading the Resources and a fixed 'cool\_down\_delay' that occurs after loading the Resources.

Note that delay Times are not affected by the efficiency of the Resources.

Extensions

REQUEST\_FIXED Extension

time - - *a Time field of model BASIC\_DELAYED*

The fixed additional Time that this Operation runs, beyond the computed 'time\_per' unit. Note that this is standard time which will be multiplied by the loaded Resources' efficiencies to determine actual time.

Default: 0

time\_per - - *a Time field of model BASIC\_DELAYED*

The standard (100% efficiency) Time per unit of this Operation that this Operation runs. Note that this is standard time which will be multiplied by the loaded Resources' efficiencies to determine actual time.

Default: 0

pre\_load\_delay - - *a Time field of model BASIC\_DELAYED*

The fixed delay Time from the start of the Operation to the time when it begins loading the Resources and running. This time occurs immediately before 'time' and 'time\_per'.

Default: 0

post\_load\_delay - - *a Time field of model BASIC\_DELAYED*

The fixed delay Time from when the Operation has finished loading the Resources until the Operation ends. This time occurs immediately after 'time' and 'time\_per'.

Default: 0

REQUEST\_FIXED -- *a process extension of model Operation*

A REQUEST\_FIXED Operation places a Request for an Item 'item' at another Site ('supplier'). The Operation requires 'time' from completion of delivery of the 'item' by the 'supplier' to completion of this Operation (when the output flows will produce to those Buffers). Thus, this could model a simplistic pickup requirement (the "delivery" is made to the 'supplier's dock from where you can pick it up). Or it could represent the time to perform inspection or receiving activity. Or it could be a simple pad on the due date given the supplier.

The REQUEST\_FIXED model has fields that references these models :  
Configuration, Item, Site, Seller.

time - - *a Time field of model REQUEST\_FIXED*

The fixed Time that this Operation requires following delivery by the 'supplier'.

Default: 0

Extensions

REQUEST\_FIXED Extension

order\_lead\_time - - *a Time field of model REQUEST\_FIXED*

Obsolete! The field 'order\_lead\_time' is functionally covered by the 'release\_fence'. To model a supplier lead time of 5 days, use a 5-day Horizon. Date for the 'release\_fence'. In that way, an unreleased REQUEST\_FIXED Operation will not be planned inside the supplier lead time. Note that the 'release\_fence', being a Horizon\_Date, allows you to specify a varying supplier lead time, more accurately modeling the real constraint. And it does so with the same mechanism that is used for non-REQUEST\_FIXED Operations.

Default: 0

Properties: obsolete=True

item - - *a Item field of model REQUEST\_FIXED*

The Item from 'site' that is being requested (same as 'configuration.item').

To set this you must have an Item. If you have just the name of an Item, then you must also have the Site. With those two you can set this to 'site.items.find(item\_name)'. However, it is generally simpler to set 'supplier' to that site and 'item\_name' to that name.

Default: [unspecified]

supplier - - *a Site field of model REQUEST\_FIXED*

The Site from which the Item is requested (same as 'item.owner').

However, it is also sellable. Selling this will cause 'configuration.item' to be set to the Item by the same name in the newly set Site. If the new Site does not have an Item with that name, then it is set to the [unspecified] Item at that Site.

Note that selling 'item' or 'configuration.item' to an Item also sets this 'supplier' to be the Site that owns that Item.

Default: configuration.item.owner

item\_name - - *a Symbol field of model REQUEST\_FIXED*

This is the same as 'item.name', however it is also sellable. Selling 'item\_name' indirectly sets 'configuration.item' to the Item with the specified name in the same Site ('supplier'). If there is no such Item, a Field\_Error is reported and nothing is changed.

Default: configuration.item.name

seller - - *a Seller field of model REQUEST\_FIXED*

The Seller through which this Site purchases from 'supplier'. If this is left [unspecified], then the price will be assumed zero, and no forecasted demand will be consumed at the 'supplier'.

Default: [unspecified]

Extensions	REQUEST_FIXED_WITH_ANALYSIS Extension
------------	---------------------------------------

**REQUEST\_FIXED\_WITH\_ANALYSIS -- a process extension of model Operation**

A REQUEST\_FIXED\_WITH\_ANALYSIS Operation places a Request for an Item 'item' at another Site ('supplier'). The Operation requires time 'from completion of delivery of the item' by the 'supplier' to completion of this Operation (when the output Flows will supply to those Buffers). Thus, this could model a simplistic pickup requirement (the "delivery" is made to the 'supplier's dock from where you can pick it up). Or it could represent the time to perform inspection or receiving activity. Or it could be a simple pad on the due date given the 'supplier'. Additionally, when the corresponding item request is created, \*\*\* add description \*\*\*

The REQUEST\_FIXED\_WITH\_ANALYSIS model has fields that references these models :

Configuration, Item, Site, Seller.

time - - a Time field of model REQUEST\_FIXED\_WITH\_ANALYSIS

The fixed Time that this Operation requires following delivery by the 'supplier'.

Default: 0

order\_lead\_time - - a Time field of model

REQUEST\_FIXED\_WITH\_ANALYSIS

The typical order lead Time that this Request requires.

Default: 0

item - - a Item field of model REQUEST\_FIXED\_WITH\_ANALYSIS  
The Item from 'site' that is being requested (same as 'configuration.item').

To set this you must have an Item. If you have just the name of an Item, then you must also have the Site. With those two you can set this to 'site.items.find(item\_name)'. However, it is generally simpler to set 'supplier' to that site and 'item\_name' to that name.  
Default: [unspecified]

supplier - - a Site field of model REQUEST\_FIXED\_WITH\_ANALYSIS  
The Site from which the Item is requested (same as 'item.owner').

However, it is also settable. Setting this will cause 'configuration.item' to be set to the Item by the same name in the newly set Site. If the new Site does not have an Item with that name, then it is set to the [unspecified] Item at that Site.

Extensions	ROUTING Extension
------------	-------------------

Note that setting 'item' or 'configuration.item' to an Item also sets this 'supplier' to be the Site that owns that Item.

Default: configuration.item.owner

item\_name - - a Symbol field of model REQUEST\_FIXED\_WITH\_ANALYSIS  
This is the same as 'item.name', however it is also settable. Setting 'item\_name' indirectly sets 'configuration.item' to the Item with the specified name in the same Site ('supplier'). If there is no such Item, a Field\_Error is reported and nothing is changed.  
Default: configuration.item.name

seller - - a Seller field of model REQUEST\_FIXED\_WITH\_ANALYSIS

The Seller through which this Site purchases from 'supplier'. If this is left [unspecified], then the price will be assumed zero, and no forecasted demand will be consumed at the 'supplier'.  
Default: [unspecified]

**ROUTING -- a process extension of model Operation**

A ROUTING Operation executes the 'sub\_operations' in order, with no overlap. The 'sub\_operations' and their ordering are defined by the 'routing\_operations'.

It can add Loads which consume Resources for the duration of the 'sub\_operations', or it can add Flows which consume Items at the beginning of the 'sub\_operations' or produce Items at the end.

If Resources are loaded by this ROUTING Operation, then the efficiency of the Resource affects all 'sub\_operations', but does not affect the time between 'sub\_operations'. For example, if a Resource used by this ROUTING Operation drops to 50%, then the maximum efficiency of any sub Operation will be 50% (if it was at 100%, the load time required will double).

If a ROUTING Operation is not 'interruptible', then the Resources it loads will be reserved for the entire duration of the routing, whether or not any 'sub\_operations' are being performed. The Resources must be available for that whole time (no 0% efficiency). This is often used for vats, molds, dies, mandrel, or other Resources that must move with the material through several Operations before being separated.

If a ROUTING Operation is 'interruptible', then the Resources it loads will only be loaded while 'sub\_operations' are using it. The Resource can be unavailable at other times. Effectively, such Loads become additional Loads of the 'sub\_operations'. This is often used to model a sequence of Operations that can choose among alternate Resources, but must all choose the same alternate.



Extensions	ROUTING Extension
------------	-------------------

Currently, 'interruptible' is not implemented, and all ROUTING operations are planned as though 'interruptible' is true.

Note that for a non-interruptible ROUTING Operation, a drop in efficiency on a Resource it loads may cause all the 'sub\_operations' to increase in time, but the total time that the Resource is reserved may not change since the time between those 'sub\_operations' may decrease.

The ROUTING model has these submodels :  
Routing\_Operation.

The ROUTING model has fields that references these models :  
Routing\_Operation.

routing\_operations - - *a list of Routing\_Operation submodels of model ROUTING*  
This defines the 'sub\_operations' of this Operation and their numbered sequence.

Extensions	Load Extensions
------------	-----------------

### 10.7 Load Extensions

10.7.1 usage\_policy extensions of model Load

RESOURCE -- *a usage\_policy extension of model Load*

A RESOURCE usage\_policy Load utilizes the specified 'resource'.

ONE -- *a usage\_policy extension of model Load*

A ONE usage\_policy Load utilizes exactly one of the Resources of the specified 'skill'.

<i>Extensions</i>	<i>Load_Size Extensions</i>
-------------------	-----------------------------

### 10.8 Load\_Size Extensions

10.8.1 load\_size\_usage extensions of model Load\_Size

**FIXED** -- *a load\_size\_usage extension of model Load\_Size*

Consumes same amount of capacity independent of the number of units of operation\_plan.

*fixed\_quantity* -- *a Quantity field of model FIXED*  
The fixed quantity of capacity which is consumed.  
Default: 0

**LINEAR** -- *a load\_size\_usage extension of model Load\_Size*

Capacity consumption increases as number of units of operation\_plan For example, weight

*linear\_quantity* -- *a Quantity field of model LINEAR*  
The per\_unit capacity that is consumed.  
Default: 0

<i>Extensions</i>	<i>Flow Extensions</i>
-------------------	------------------------

### 10.9 Flow Extensions

10.9.1 usage\_policy extensions of model Flow

**CONSUME\_FIXED** -- *a usage\_policy extension of model Flow*

A CONSUME\_FIXED Flow consumes a fixed 'quantity' of the Item, independent of the number of units of the Operation. For example, you may need to fill the tank with 10 gallons of solvent, no matter how many boards will be dipped.

For compatibility with older systems that do not allow multiple PRODUCED\_FIXED Flows, a negative quantity indicates that this is produced rather than consumed.

*fixed\_quantity* -- *a Quantity field of model CONSUME\_FIXED*  
The fixed Quantity of the Item that is produced or consumed by this Operation, independent of the number of units of this Operation performed. An example would be an Operation that requires 100 gallons of plating solution to be put in the tank, independent of the number of boards to be processed by the Operation.  
Default: 0

**CONSUME\_PER** -- *a usage\_policy extension of model Flow*

A CONSUME\_PER Flow consumes 'quantity\_per' of the Item per unit of the Operation.

For compatibility with older systems that do not allow multiple PRODUCED\_PER Flows, a negative quantity indicates that this is produced rather than consumed.

*quantity\_per* -- *a Quantity field of model CONSUME\_PER*  
The Quantity of this Item that is consumed per unit of the Operation.  
Default: 1

**PRODUCE\_FIXED** -- *a usage\_policy extension of model Flow*

A PRODUCE\_FIXED Flow produces 'quantity' of the Item, independent of the number of units of the Operation being performed. For example, 10 gallons of waste solvent may be used up and "produced", no matter how many boards were dipped in the solvent.

Extensions	PRODUCE_PER Extension
------------	-----------------------

**fixed\_quantity** - - *a Quantity field of model PRODUCE\_FIXED*  
 The fixed Quantity of the Item that is produced by this Operation, independent of the number of units of this Operation performed. An example would be an Operation that requires 100 gallons of plating solution to be put in the tank, independent of the number of boards to be processed by the Operation.  
 Default: 0

**PRODUCE\_PER** - - *a usage\_policy extension of model Flow*

A PRODUCE\_PER Flow produces 'quantity\_per' of the Item per unit of the Operation.

**quantity\_per** - - *a Quantity field of model PRODUCE\_PER*  
 The Quantity of this Item that is produced per unit of the Operation.  
 Default: 1

**PRODUCE\_YIELD** - - *a usage\_policy extension of model Flow*

The PRODUCE\_YIELD Flow produces 'quantity\_per' of the Item per unit of the Operation, less lossage due to less than 100% yield. Yield is specified as a fixed average value, 'yield', for long-term planning, and a typical worst-case value, 'yield\_near', for near-term planning (as specified by 'near\_time'). For yield values that can vary over time, see the PRODUCE\_YIELD\_CALENDAR usage\_policy extension.

**quantity\_per** - - *a Quantity field of model PRODUCE\_YIELD*  
 The Quantity of this Item that is produced or consumed per unit of the Operation.  
 Default: 1

**yield** - - *a Percentage field of model PRODUCE\_YIELD*  
 The expected yield of this Item from this Operation. This should be in the range 0% < yield <= 100%.  
 Default: 100%

**yield\_near** - - *a Percentage field of model PRODUCE\_YIELD*  
 The worst-case yield that should be used in near-term calculations. This should be in the range 0% < yield\_near <= 100%.  
 Default: 100%

**near\_time** - - *a Time field of model PRODUCE\_YIELD*  
 The time fence defining "near-term" for use of yield\_near rather than yield. The purpose is to plan for worst-case yields in the near term, since there is no time to recover. Note that if you planned for worst-case yields over the entire horizon, you may significantly overplan.

Extensions	PRODUCE_YIELD_CALENDAR Extension
------------	----------------------------------

Default: 0

**PRODUCE\_YIELD\_CALENDAR** - - *a usage\_policy extension of model Flow*

The PRODUCE\_YIELD\_CALENDAR Flow produces 'quantity\_per' of the Item per unit of the Operation, less lossage due to less than 100% yield.

The yield is specified much like PRODUCE\_YIELD: there is an average yield value, 'yield', for long-term planning, and a typical worst-case value, 'yield\_near', for near-term planning (as specified by 'near\_time'). However, there is also a 'calendar' which specifies multipliers for those yield numbers. The yield changes instantaneously at each Calendar entry. (For linear ramping between entries, see PRODUCE\_YIELD\_RAMP\_CALENDAR.)

This is commonly used to reflect process or machine changes that improve yields.

The PRODUCE\_YIELD\_CALENDAR model has fields that references these models:  
 Calendar.

**quantity\_per** - - *a Quantity field of model PRODUCE\_YIELD\_CALENDAR*  
 The Quantity of this Item that is produced or consumed per unit of the Operation.  
 Default: 1

**yield** - - *a Percentage field of model PRODUCE\_YIELD\_CALENDAR*  
 The base expected yield of this Item from this Operation. This will be multiplied by the appropriate value from the "calendar".  
 Default: 100%

**yield\_near** - - *a Percentage field of model PRODUCE\_YIELD\_CALENDAR*  
 The base worst-case yield that should be used in near-term calculations. This will be multiplied by the appropriate value from the "calendar".  
 Default: 100%

**near\_time** - - *a Time field of model PRODUCE\_YIELD\_CALENDAR*  
 The time fence defining "near-term" for use of yield\_near rather than yield. The purpose is to plan for worst-case yields in the near term, since there is no time to recover. Note that if you planned for worst-case yields over the entire horizon, you may significantly overplan.  
 Default: 0

Extensions	PRODUCE_YIELD_CALENDAR Extension
------------	----------------------------------

calendar - - *a* Calendar *field of model* PRODUCE\_YIELD\_CALENDAR  
 Calendar of factors that are multiplied with the base\_yields to give the yield to use in  
 planning.  
 Default: [unspecified]

Extensions	Operation_Problem_Detector Extensions
------------	---------------------------------------

10.10 Operation\_Problem\_Detector Extensions  
 10.10.1 detector extensions of model Operation\_Problem\_Detector

## 10.11 Operation\_Plan Extensions

### 10.11.1 motive extensions of model Operation\_Plan

#### PRODUCE -- a motive extension of model Operation\_Plan

A PRODUCE motive Operation\_Plan is performed in order to produce Items with the specified into *buffer\_plan* a Quantity between 'min' and 'max' into *buffer\_plan* within the 'due' *Date\_Range* (neither before, nor after). The 'configuration' specifies any details about the Item to be produced, according to the Item's 'spec' fields.

The PRODUCE model has fields that references these models :  
*Buffer\_Plan*, *Configuration*.

*buffer\_plan* - - a *Buffer\_Plan* field of model PRODUCE  
The *Buffer\_Plan* into which this Operation\_Plan should produce the Items. Note that this is the *Buffer\_Plan* that planned this Operation\_Plan.  
Properties: *Export-Only Field*

*configuration* - - a *Configuration* field of model PRODUCE  
The desired characteristics of the Items that this Operation\_Plan should produce into *buffer\_plan*.  
Default: [unspecified]  
Properties: *Export-Only Field*

#### CONSUME -- a motive extension of model Operation\_Plan

A CONSUME motive Operation\_Plan is performed in order to consume a Quantity between 'min' and 'max' from *buffer\_plan* within the 'due' *Date\_Range* (neither before, nor after). The 'configuration' specifies any details about the Item to be consumed, according to the Item's 'spec' fields.

The CONSUME model has fields that references these models :  
*Buffer\_Plan*, *Configuration*.

*buffer\_plan* - - a *Buffer\_Plan* field of model CONSUME  
The *Buffer\_Plan* from which this Operation\_Plan should consume the Items. Note that this is the *Buffer\_Plan* that planned this Operation\_Plan.  
Properties: *Export-Only Field*

*configuration* - - a *Configuration* field of model CONSUME  
The desired characteristics of the Items that this Operation\_Plan should consume from *buffer\_plan*.  
Default: [unspecified]  
Properties: *Export-Only Field*

#### DELIVER -- a motive extension of model Operation\_Plan

A DELIVER motive Operation\_Plan is performed in order to deliver the a Quantity between 'min' and 'max' from a source *Buffer* to a specified *Location* (not a *Buffer*). The 'configuration' specifies any details about the Item to be consumed, according to the Item's 'spec' fields.

The DELIVER model has fields that references these models :  
*Item\_Request*, *Item\_Promise*, *Configuration*.

*motive\_request* - - a *Item\_Request* field of model DELIVER  
The *Item\_Request* for which this Operation may be planned to satisfy. This is the same as '*motive\_promise.item\_request*'.

Note that this Operation\_Plan may not be trying to satisfy this '*motive\_request*'; if '*planned\_for\_promise*' is "true", then this Operation\_Plan is planned to satisfy the '*motive\_promise*' rather than this '*motive\_request*'.  
Properties: *Export-Only Field*

*motive\_promise* - - a *Item\_Promise* field of model DELIVER  
The *Item\_Promise* for which this Operation may be planned to satisfy. This is the same as '*motive\_request.item\_promise*'.

Note that this Operation\_Plan may not be trying to satisfy this '*motive\_promise*'; if '*planned\_for\_promise*' is "false", then this Operation\_Plan is planned to satisfy the '*motive\_request*' rather than this '*motive\_promise*'.  
Properties: *Export-Only Field*

*planned\_for\_promise* - - a *Logical* field of model DELIVER  
If "true", then this Operation\_Plan has been planned to satisfy the '*motive\_promise*'; otherwise, it is planned to satisfy the '*motive\_request*'. Setting this will cause this Operation\_Plan to be replanned accordingly. This can also be set by functions such as '*plan\_to\_satisfy*' on the '*motive\_request*' or the '*motive\_promise*'.

Note that the Problems that are detected depend upon how this is set. For example, PROMISE\_PLANNED\_LATE will only be detected if this is "true". Conversely, REQUEST\_PLANNED\_LATE will only be detected if this is "false".

Default: false  
Properties: Export-Only Field

**Item -- a Item field of model DELIVER**

The Item which should be delivered to the customer of the Item\_Request.  
Properties: Export-Only Field

**configuration -- a Configuration field of model DELIVER**

The desired characteristics of the Items that this Operation\_Plan should deliver to the 'destination'.

Default: [unspecified]  
Properties: Export-Only Field

**RECEIVE -- a motive extension of model Operation\_Plan**

A RECEIVE motive Operation\_Plan is performed in order to receive the offered quantity from an unsolicited promise. The Item specifies which buffer to place the quantity in.

The RECEIVE model has fields that references these models :  
Item.

**Item -- a Item field of model RECEIVE**

The Item which is being received  
Properties: Export-Only Field

**10.11.2 process extensions of model Operation\_Plan**

**ALTERNATES\_PRIMARY -- a process extension of model Operation\_Plan**

An ALTERNATES\_PRIMARY Operation executes one of the sub\_operations. It defaults to using the primary Operation, the one with the largest 'percentage'. When it needs to offload, it selects an alternate probabilistically, selecting proportional to 'percentage'.

The difference between ALTERNATES\_PRIMARY and ALTERNATES\_PROPORTIONAL is that ALTERNATES\_PRIMARY always selects the one with the largest 'percentage' initially, but ALTERNATES\_PROPORTIONAL makes the initial selection probabilistically also.

move\_to\_alternate (Operation\_Plan, Operation, Quantity), move\_to\_alternate (Operation\_Plan, Operation) , move\_to\_alternate (Operation\_Plan) ,  
move\_to\_alternate -- a Void field of model ALTERNATES\_PRIMARY

Causes the specified sub\_operation\_plan to change to the specified alternate (if none specified, then it chooses for you). If no arguments are given, it attempts to alternate itself (ie find the alternate whom it is underneath, and call move\_to\_alternate on it).

Properties: command=True Export-Only Field

**ALTERNATES\_PROPORTIONAL -- a process extension of model Operation\_Plan**

An ALTERNATES\_PROPORTIONAL Operation executes one of the sub\_operations. It chooses which alternate to use probabilistically, weighed by the Alternate\_Operation's percentage.

move\_to\_alternate (Operation\_Plan, Operation, Quantity), move\_to\_alternate (Operation\_Plan, Operation) , move\_to\_alternate (Operation\_Plan) ,  
move\_to\_alternate -- a Void field of model ALTERNATES\_PROPORTIONAL

Causes the specified sub\_operation\_plan to change to the specified alternate (if none specified, then it chooses for you). If no arguments are given, it attempts to alternate itself (ie find the alternate whom it is underneath, and call move\_to\_alternate on it).

Properties: command=True Export-Only Field

**EFFECTIVE\_CALENDAR -- a process extension of model Operation\_Plan**

An EFFECTIVE\_CALENDAR Operation executes one of the sub\_operations. Selection of the sub operation will be done based on the effectivity specified by the calendar for that sub operation. Each sub operation can specify their own effectivity calendar. If more than one sub operation is effective on a given date, probabilistic selection will be done based on the 'percentage' value.

move\_to\_alternate (Operation\_Plan, Operation, Quantity), move\_to\_alternate (Operation\_Plan, Operation) , move\_to\_alternate (Operation\_Plan) ,  
move\_to\_alternate -- a Void field of model EFFECTIVE\_CALENDAR

Causes the specified sub\_operation\_plan to change to the specified alternate (if none specified, then it chooses for you). If no arguments are given, it attempts to alternate itself (ie find the alternate whom it is underneath, and call move\_to\_alternate on it).

Properties: command=True Export-Only Field

**DELAY\_ONLY\_FIXED -- a process extension of model Operation\_Plan**

A DELAY\_ONLY\_FIXED Operation loads no Resource. It simply runs for a fixed amount of time, independent of the quantity being processed. Thus, this can be used to insert a fixed delay in the processing.

**DELAY\_ONLY\_BASIC -- a process extension of model Operation\_Plan**

A DELAY\_ONLY\_BASIC Operation loads no Resource. It simply runs for a time equal to `fixed_time + units*time_per`. Thus, this can be used to insert a per-unit delay in the processing.

**BASIC\_CALENDARS -- a process extension of model Operation\_Plan**

A BASIC\_CALENDARS Operation loads the Resources for a time equal to `time + units*time_per/resource_efficiency`, where both the time per unit of the operation (`time_per`) and the additional operation time (`time`) are specified using calendars.

**FIXED\_TIME -- a process extension of model Operation\_Plan**

A FIXED\_TIME Operation loads a Resource for a fixed amount of Time, independent of the Quantity being processed.

**TIME\_MULTIPLE -- a process extension of model Operation\_Plan**

A TIME\_MULTIPLE Operation loads a Resource for a Time that increases in discrete multiples of `base_time`, based on the number of units as a multiple of `base_units`. More mathematically, the time is `round_up(units / base_units) * base_time`.

**BASIC -- a process extension of model Operation\_Plan**

A BASIC Operation loads the Resources for a time equal to `time + units*time_per/resource_efficiency`.

**BASIC\_DELAYED -- a process extension of model Operation\_Plan**

A BASIC\_DELAYED Operation loads the Resources for a time equal to `time + units*time_per/resource_efficiency`. In addition, the BASIC\_DELAYED Operation total time includes a fixed `warm_up_delay` that occurs prior to loading the Resources and a fixed `cool_down_delay` that occurs after loading the Resources.

Note that delay Times are not affected by the efficiency of the Resources.

**REQUEST\_FIXED -- a process extension of model Operation\_Plan**

A REQUEST\_FIXED Operation places a Request for an Item 'item' at another Site (supplier). The Operation requires 'time' from completion of delivery of the 'item' by the 'supplier' to completion of this Operation (when the output Flows will produce to those Buffers).

The REQUEST\_FIXED model has fields that references these models :  
Item\_Request.

Item\_request -- a Item\_Request field of model REQUEST\_FIXED  
The Item\_Request which is generated by this Operation\_Plan.  
Properties: Export-Only Field

**REQUEST\_FIXED\_WITH\_ANALYSIS -- a process extension of model Operation\_Plan**

A REQUEST\_FIXED\_WITH\_ANALYSIS Operation places a Request for an Item 'item' at another Site (supplier). The Operation requires 'time' from completion of delivery of the 'item' by the 'supplier' to completion of this Operation (when the output Flows will supply to those Buffers).

The REQUEST\_FIXED\_WITH\_ANALYSIS model has fields that references these models :  
Item\_Request.

Item\_request -- a Item\_Request field of model  
REQUEST\_FIXED\_WITH\_ANALYSIS  
The Item\_Request which is generated by this Operation\_Plan.  
Properties: Export-Only Field

**ROUTING -- a process extension of model Operation\_Plan**

A ROUTING Operation executes the 'sub\_operations' in order, with no overlap. The 'sub\_operations' and their ordering are defined by the 'routing\_operations'.

It can add Loads which consume Resources for the duration of the 'sub\_operations', or it can add Flows which consume Items at the beginning of the 'sub\_operations' or produce Items at the end.

Note that a ROUTING Operation does not have any 'sid\_time' of its own (it's always "0"). If Resources are loaded by this ROUTING Operation, then the efficiency of the Resource affects all 'sub\_operations', but does not affect the time between sub\_operations. For example, if a Resource used by this ROUTING Operation drops to 50%, then the maximum efficiency of any sub Operation will be 50% (if it was at 100%, the load time required will double).

## 10.12 Resource Extensions

### 10.12.1 efficiency extensions of model Resource

#### FIXED -- a efficiency extension of model Resource

A FIXED efficiency Resource has a fixed efficiency level at all times. The default value is the very common case of 100%.

**fixed\_efficiency** -- a Percentage field of model FIXED

The efficiency level (how fast it is processing).

Default: 100%

#### CALENDAR -- a efficiency extension of model Resource

A CALENDAR efficiency Resource has efficiency that will vary over time as specified in a Calendar. This may be used to model a shift calendar, with potentially different efficiency in different shifts and downtimes. It may also be used to represent any efficiency with cyclic behavior relative to the calendar. Or it may be used to model in detail a ramp up, ramp down, or engineering change to become effective.

Further, the options for increasing efficiency by incurring additional cost can be modeled. Calendar\_Entry's with a non-zero 'charge' are treated as options for increasing efficiency when needed.

The CALENDAR model has fields that references these models :  
Calendar.

**efficiency\_calendar** -- a Calendar field of model CALENDAR

A Calendar with PERCENTAGE 'entries', where the Percentage is the efficiency level at those dates. Entries with a non-zero 'charge' are treated as options for increasing efficiency when needed.

Default: [unspecified]

### 10.12.2 variability extensions of model Resource

#### ZERO -- a variability extension of model Resource

A ZERO variability Resource does not have significant enough variability to need planning compensation or padding.

#### FIXED -- a variability extension of model Resource



Extensions	maintenance extensions of model Resource
------------	--

A **FIXED** variability Resource has variability that is compensated for by padding the arrival of upstream Operations and the departure to downstream Operations by fixed Times.

The `downstream_pad` reduces the effects that the variability of this Resource has upon the downstream Resources. So, if this machine goes down for 1 hour every 10 days, you may want to set `downstream_pad` to 1 hour. In that way, all downstream Operations and Buffers are expecting to receive the material an hour late -- and thus, their schedules are not disturbed when the machine goes down for an hour.

The `upstream_pad` prevents this Resource from starving when it runs faster than expected, or allows it to run things in a different order by providing it a certain level of WIP in front of it. If this machine sometimes gets as much as an hour ahead of schedule, you may want to set `upstream_pad` to 1 hour. In that way, all upstream Operations and Buffers plan to produce the material 1 hour early, and thus the machine does not starve when it gets an hour ahead.

Note that increasing these pads will increase WIP on the floor. Thus, the pads should generally be kept as small as possible without making the plan too brittle in the face of disturbances. The `upstream_pad` may not be important on any but the most utilized Resources (the ones where you do not want it to ever starve).

**upstream\_pad** -- a Time field of model **FIXED**  
The extra Time by which to pad the arrival from upstream Operations to Operations on this Resource.  
Default: 0

**downstream\_pad** -- a Time field of model **FIXED**  
The extra Time by which to pad the departure to downstream Operations from Operations on this Resource.  
Default: 0

10.12.3 maintenance extensions of model Resource

**ZERO** -- a maintenance extension of model Resource

A **ZERO** maintenance Resource does not need any maintenance.

10.12.4 size extensions of model Resource

**UNLIMITED** -- a size extension of model Resource

Extensions	FIXED_COUNT Extension
------------	-----------------------

An **UNLIMITED** size Resource has no relevant size constraints. Although physically unreasonable, this is very common for planning purposes. Most Operations are designed to fit on the Resources they use, and thus cannot be "too big". All Operations are considered to have size of "1".

**FIXED\_COUNT** -- a size extension of model Resource

A **FIXED\_COUNT** size Resource has a single fixed size limit at all times. The size of an Operation\_Plan is "1". Thus, the `max_operation_count` is the maximum number of Operation\_Plans that can run on the Resource at once. This is commonly used to model several identical Resources as one Resource. The `max_operation_count` is effectively the number of identical Resources that are available.

The default value is the common case of "oo" (infinite) which means there are no size limits.

**max\_operation\_count** -- a Integer field of model **FIXED\_COUNT**  
The maximum number of Operation\_Plans that can use this Resource at once.  
Default: oo (infinite, unlimited, unless)

**FIXED\_QUANTITY** -- a size extension of model Resource

A **FIXED\_QUANTITY** size Resource has a single fixed size limit at all times. The size of an Operation is measured by converting the units of the Operation\_Plan to the unit of Measure of the `max_size`. Thus, if the `max_size` is "500 kg", the size of an Operation\_Plan will be the units of the Operation\_Plan converted to "kg". If the `max_size` is unitless, then the number of units of the Operation\_Plan is used.

The default value is the common case of "oo" (infinite) which means there are no size limits.

**max\_size** -- a Quantity field of model **FIXED\_QUANTITY**  
The maximum total size of Operation\_Plans that can run on this Resource at once.  
Default: oo (infinite, unlimited, unless)

**CALENDAR\_COUNT** -- a size extension of model Resource

A **CALENDAR\_COUNT** size Resource can handle a maximum number of Operation\_Plans at once that varies over time as specified in a Calendar. Calendar\_Entry's with a non-zero 'charge' are treated as options for increasing size when needed.

Extensions	MULTI_DIMENSION Extension
------------	---------------------------

The size of an Operation\_Plan is "1". Thus, the Number in the Calendar is the maximum number of Operation\_Plans that can run on the Resource at once. This is commonly used to model several identical Resources as one Resource. The Number in each Calendar\_Entry is effectively the number of identical Resources that are available.

The CALENDAR\_COUNT model has fields that references these models :  
Calendar:

size\_calendar -- a Calendar field of model CALENDAR\_COUNT

A Calendar with NUMBER 'entries', where the Quantity is the maximum number of Operation\_Plans.

Default: [unspecified]

#### MULTI\_DIMENSION -- a size extension of model Resource

A MULTI\_DIMENSION size resource can model size limits in multiple dimensions. Each dimension can be declared with a min\_size and a max\_size limit.

'aggregate\_count' represents the total number of resources available. Each of these 'aggregate\_count' resources are loaded the same way using the load\_policy. For example, the EXCLUSIVE\_USE load\_policy will load 'aggregate\_count' resources, each of which is of type EXCLUSIVE\_USE. The SIMPLE\_CONSOLIDATION load\_policy will load 'aggregate\_count' resources, each of which has consolidation restriction specified by 'dimensions'.

Each resource has size restrictions as specified in 'dimensions'. The loads on this resource should have 'load\_sizes' defined in all the dimensions specified here. The load\_policy decided how to use the various dimensions. For example EXCLUSIVE\_USE, SHARED\_USE and SIMPLE\_CONSOLIDATION load\_policies consider the dimensions orthogonally. Violation in any one dimension flags problems. Future load\_policies can consider violations in only one dimension or combine them arbitrarily.

The MULTI\_DIMENSION model has these submodels :  
Size\_Dimension.

The MULTI\_DIMENSION model has fields that references these models :  
Size\_Dimension.

Extensions	load_policy extensions of model Resource
------------	--

dimensions -- a list of Size\_Dimension submodels of model MULTI\_DIMENSION  
The various dimensions of the size of each one of the aggregate resource. The dimensions need not be orthogonal.

#### 10.12.5 load\_policy extensions of model Resource

##### SIMPLE\_CONSOLIDATION -- a load\_policy extension of model Resource

An SIMPLE\_CONSOLIDATION Resource lets users consolidate the load\_plans on this resource. No loading Problems will be detected or resolved (such as two Operations planned at once).

consolidation\_fence -- a Horizon\_Date field of model  
SIMPLE\_CONSOLIDATION

The fence after which no consolidation problems are detected  
Default: 'infinite future'

##### INFINITE\_USE -- a load\_policy extension of model Resource

An INFINITE\_USE Resource is considered to have "infinite capacity" in the traditional sense. That is, each Operation is loaded appropriately (considering its capacity), but without considering any other Operations. No loading Problems will be detected or resolved (such as two Operations planned at once). But you can still view the planned load versus the actual capacity available.

##### EXCLUSIVE\_USE -- a load\_policy extension of model Resource

An EXCLUSIVE\_USE Resource can handle only one load at any given time. All overlapping loads are identified as Problems, size of the resource is ignored in planning the Resource. This is done deliberately to have fast run times for this most common model. It is an error to have a size other than UNLIMITED (the default) for this extension.

##### SHARED\_USE -- a load\_policy extension of model Resource

A SHARED\_USE Resource can handle multiple Loads at the same time. The Loads are allowed to overlap arbitrarily, as long as they do not exceed the size limits at that time as specified by the 'size' extension.

For example, if there is a FIXED\_QUANTITY size limit of "2 tons", then up to 2 tons of Loads may be simultaneously processed. If there is a FIXED\_COUNT size limit of "5", then up to 5 Loads will be allowed at once. Note that if there is a FIXED\_COUNT size limit of "1", then this is equivalent to EXCLUSIVE\_USE (just one Load at a time).

All Loads are considered compatible. In contrast, a SHARED\_USE\_SETUP Resource can only run together Operations with identical setups. A common example would be 10 identical Resources that are preferably modeled as one Resource that can run 10 things at once.

The SHARED\_USE model has fields that references these models :

Horizon.

buckets - - *a Horizon field of model SHARED\_USE*  
The Horizon that defines the 'dates' of the buckets used by this policy after the horizon\_fence' date.  
Default: none

bucket\_fence - - *a Horizon\_Date field of model SHARED\_USE*  
The horizon-relative Date beyond which this load\_policy switches to coarser planning estimates. After the fence Date, this load\_policy will do bucket based problem detection and resolution. This date is relative to the plan.start  
Default: none

min\_load - - *a Percentage field of model SHARED\_USE*  
The minimum load on this resource, expressed as a percentage of the size. The resource has to have at least 'min\_load' from Plan.start to 'min\_load\_fence'. For example, if the 'min\_load' is 80%, then this resource has to be utilized by at least 80% of its capacity from Plan.start to 'min\_load\_fence'. If the utilization goes below 80%, we flag an UNDERLOAD problem from Plan.start till 'min\_load\_fence'. The valid values of the fields are from 0% to 100%. The default value of 'min\_load' is 0%, meaning there is no restriction.  
Default: 0%

min\_load\_fence - - *a Horizon\_Date field of model SHARED\_USE*  
The relative date from Plan.start till which we do not want this resource to be underutilized.  
Default: none

max\_bucket\_load - - *a Percentage field of model SHARED\_USE*  
The resource cannot be over-utilized by more than 'max\_bucket\_load' in any bucket. This is expressed as a percentage of the size in actual hours of the resource in this bucket (resource.Plan.available\_time). For example, if the 'max\_bucket\_load' is 80%, then this resource cannot be utilized by more than 80% of its capacity in any bucket. If the total utilization goes above 80% of the total capacity, a BUCKET\_OVERSIZE problem will be flagged in that bucket. The default value of 'max\_bucket\_load' is 100%, meaning the resource cannot be loaded more than 100% of its capacity.  
Default: 100%

upstream\_bucket\_pad - - *a Time field of model SHARED\_USE*  
The extra Time by which to pad the arrival from upstream Operations to Operations on this Resource. This pad is in addition to any upstream pad imposed by the variability extension. This is the upstream pad needed to have a stable plan when switching from bucketized planning to detailed planning. This will be added to the operation\_plan only if it starts loading the resource at or after the 'bucket\_fence'.  
Default: 0

downstream\_bucket\_pad - - *a Time field of model SHARED\_USE*  
The extra Time by which to pad the departure to downstream Operations from Operations on this Resource. This pad is in addition to any downstream pad imposed by the variability extension. This is the downstream pad needed to have a stable plan when switching from bucketized planning to detailed planning. This will be added to the operation\_plan only if it ends loading the resource after the 'bucket\_fence'.  
Default: 0

10.13 Resource\_Skill Extensions

10.13.1 efficiency extensions of model Resource\_Skill

FIXED -- a efficiency extension of model Resource\_Skill

A FIXED efficiency Resource\_Skill has a fixed efficiency level at all times. The default value is the very common case of 100%.

*fixed\_efficiency* -- a Percentage field of model FIXED

The efficiency level of the resource in performing this skill  
Default: 100%

CALENDAR -- a efficiency extension of model Resource\_Skill

A CALENDAR efficiency Resource\_Skill has efficiency that will vary over time as specified in a Calendar.

Further, the options for increasing efficiency by incurring additional cost can be modeled. Calendar\_Entry's with a non-zero 'charge' are treated as options for increasing efficiency when needed. Uses the Default\_Efficiency\_Calendar, if efficiency\_calendar is not set.

The CALENDAR model has fields that references these models :  
Calendar.

*efficiency\_calendar* -- a Calendar field of model CALENDAR

A Calendar with PERCENTAGE 'entries', where the Percentage is the efficiency level at those dates. Entries with a non-zero 'charge' are treated as options for increasing efficiency when needed.  
Default: [unspecified]

10.14 Resource\_Problem\_Detector Extensions

10.14.1 detector extensions of model Resource\_Problem\_Detector

<i>Extensions</i>	<i>Resource_Plan Extensions</i>
-------------------	---------------------------------

### 10.15 Resource\_Plan Extensions

#### 10.15.1 efficiency extensions of model Resource\_Plan

#### 10.15.2 load\_policy extensions of model Resource\_Plan

#### SIMPLE\_CONSOLIDATION -- a load\_policy extension of model Resource\_Plan

An SIMPLE\_CONSOLIDATION Resource lets users consolidate the load\_plans on this resource. No load Problems will be detected or resolved (such as two Operations planned at once).

The SIMPLE\_CONSOLIDATION model has these submodels :

Consolidation.

The SIMPLE\_CONSOLIDATION model has fields that references these models :

Consolidation.

consolidations -- a list of Consolidation submodels of model  
SIMPLE\_CONSOLIDATION  
'consolidations' are created to group load\_plans together.

unconsolidated\_operation\_plans -- a List(Operation\_Plan) field of model  
SIMPLE\_CONSOLIDATION

The operation\_plans that are planned on this resource but do not belong to any consolidation

Properties: Export-Only Field

#### INFINITE\_USE -- a load\_policy extension of model Resource\_Plan

No loading Problems will be detected or resolved (such as two Operations planned at once). But you can still view the planned load versus the actual capacity available.

#### EXCLUSIVE\_USE -- a load\_policy extension of model Resource\_Plan

All overlapping loads are identified as Problems. size of the resource is ignored in planning the Resource. This is done deliberately to have fast run times for this most common model.

#### SHARED\_USE -- a load\_policy extension of model Resource\_Plan

<i>Extensions</i>	<i>size extensions of model Resource_Plan</i>
-------------------	---

A SHARED\_USE Resource can handle multiple Loads at the same time. The Loads are allowed to overlap arbitrarily, as long as they do not exceed the size limits at that time as specified by the 'size' extension.

For example, if there is a FIXED\_QUANTITY size limit of "2 tons", then up to 2 tons of Loads may be simultaneously processed. If there is a FIXED\_COUNT size limit of "5", then up to 5 Loads will be allowed at once. Note that if there is a FIXED\_COUNT size limit of "1", then this is equivalent to EXCLUSIVE\_USE (just one Load at a time).

All Loads are considered compatible. In contrast, a SHARED\_USE\_SETUP Resource can only run together Operations with identical setups. A common example would be 10 identical Resources that are preferably modeled as one Resource that can run 10 things at once.

#### 10.15.3 size extensions of model Resource\_Plan

#### 10.15.4 maintenance extensions of model Resource\_Plan

Extensions	Buffer Extensions
------------	-------------------

### 10.16 Buffer Extensions

#### 10.16.1 Flow\_policy extensions of model Buffer

##### BUCKETED\_NESTED\_SORT -- a Flow\_policy extension of model Buffer

A BUCKETED\_NESTED\_SORT Buffer manages the Flow\_Plans in buckets of time, as specified by the 'horizon'.

Within each bucket, the consuming Flow\_Plans are sorted by doing a nested sort on one or more 'criteria': the primary sort key (the highest 'ranking 'criteria') is used first, the second criteria is used to break ties in the first sort, and so on. Within each bucket, the producing Flow\_Plans generated by this Buffer can be broken out according to a subset of the 'criteria' that are being used.

A target 'ending\_on\_hand' will be maintained at the end of the bucket, as computed from the sum of a fixed Quantity and a percentage of the flow in the next bucket.

The BUCKETED\_NESTED\_SORT flow\_policy should preferably be used on end Item Buffers (buffers whose consuming operation plans have a DELIVERY motive). Many of the sort criteria function well only if this is an end Item Buffer, though (if the delivery Operations for the Requestis consume directly from this Buffer). In the future we will provide mechanisms for extending the visibility of Request properties beyond the end Item Buffers. See the Flow\_Criterion extensions for more information on their behavior.

The BUCKETED\_NESTED\_SORT model has these submodels:

Flow\_Criterion.

The BUCKETED\_NESTED\_SORT model has fields that references these models:

Operation, Horizon, Product\_Root, Flow\_Criterion.

producing\_operation -- a Operation field of model

BUCKETED\_NESTED\_SORT

The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field\_Error for this field to be [unspecified], the default, if any Buffer\_Plan needs to issue a producing Operation\_Plan.

Default: [unspecified]

Extensions	BUCKETED_NESTED_SORT Extension
------------	--------------------------------

horizon -- a Horizon field of model BUCKETED\_NESTED\_SORT

The Horizon that defines the 'dates' of the 'buckets' used by this policy.

Default: [unspecified]

fixed\_ending\_on\_hand -- a Quantity field of model

BUCKETED\_NESTED\_SORT

The on\_hand to be present at the end of each bucket should be the sum of this 'fixed\_ending\_on\_hand' and the 'per\_next\_ending\_on\_hand' multiplied by the consuming\_flow from the next bucket.

The net target value for each bucket in a particular Plan is available in the 'ending\_on\_hand' field of the 'priority\_buckets' of the Buffer\_Plan. That computed value is also settable, allowing the user to override the target 'ending\_on\_hand' for a bucket without modifying these Buffer fields.

This Quantity will be converted into the unit of this Buffer.

Default: 0

per\_next\_ending\_on\_hand -- a Percentage field of model

BUCKETED\_NESTED\_SORT

The on\_hand to be present at the end of each bucket should be the sum of the 'fixed\_ending\_on\_hand' and this 'per\_next\_ending\_on\_hand' multiplied by the consuming\_flow from the next bucket. If there is no next bucket, then this bucket is used.

The net target value for each bucket in a particular Plan is available in the 'ending\_on\_hand' field of the 'priority\_buckets' of the Buffer\_Plan. That computed value is also settable, allowing the user to override the target 'ending\_on\_hand' for a bucket without modifying these Buffer fields.

Default: 0%

default\_product\_root -- a Product\_Root field of model

BUCKETED\_NESTED\_SORT

When this field is specified, EXCESS\_ON\_HAND problems are resolved by creating forecast promises to the 'default\_product\_root'. In this way the Bucketed\_Nested\_Sort policy ensures that all excess material is allocated to the seller hierarchy instead of sitting in the buffer. Note that for the default value of this field, EXCESS\_ON\_HAND problems are not resolved.

Default: [unspecified]

do\_not\_move\_out\_forecasts - - a logical field of model  
BUCKETED\_NESTED\_SORT  
Defaults to true, when resolving Negative\_On\_Hand problems, don't move out forecast consumers. They will fall at the end of the current bucket and should not be moved into the next bucket, rather they should be shorted.  
Default: true

split\_multiple - - a Quantity field of model  
BUCKETED\_NESTED\_SORT  
Splits made to Operation\_Plans by a BUCKETED\_NESTED\_SORT Buffer will always be in an amount that is a multiple of this value. This Quantity will be converted into the unit of this Buffer.  
Default: 1  
Properties: obsolete=True

criteria - - a list of Flow\_Criterion submodels of model  
BUCKETED\_NESTED\_SORT

PRODUCING\_FLOW\_CALENDAR -- a flow\_policy extension of model Buffer

A PRODUCING\_FLOW\_CALENDAR Buffer is replenished on a set schedule specified by the 'calendar'. This Calendar based replenishments are only observed during the period specified via 'fence'. This flow\_policy will behave like MULTIPLE or BASIC(if the 'after\_fence\_quantity\_range') at and after the 'fence'. Buffer can plan a producing operation on any date as needed by the consuming Operation\_Plans.

In conjunction with Calendar based replenishments, this flow\_policy will allow users to specify lot sizing restrictions in terms of min, max and multiple. The quantity of each replenishment order is in multiples of 'multiple\_quantity' and is bounded by 'quantity\_range'.

Lot sizing restrictions can be relaxed by specifying different  
'after\_fence\_multiple\_quantity' and 'after\_fence\_quantity\_range' at and beyond the 'fence'. This Horizon\_Date 'fence' is relative to plan.start and will move with it.

This Buffer will also maintain material in quantity greater than or equal to 'min\_on\_hand' and less than or equal to 'excess\_on\_hand' in any given period. Both of these quantities can be date effective and specified via set of Date\_Ranges. By setting 'excess\_on\_hand' larger than zero, planning instability is reduced, and planning speed is increased. However, keeping it smaller minimizes excess inventory.

User can also specify date effective 'min\_time' which gives a fixed time of consumption that should always be covered (for example, setting 'min\_time' to "3 weeks" will keep enough stock to cover the next 3 weeks of out flows).

The PRODUCING\_FLOW\_CALENDAR model has fields that references these models :  
Calendar, Operation.

calendar - - a Calendar field of model  
PRODUCING\_FLOW\_CALENDAR  
A Calendar of entries that define the preset schedule of replenishments. Each Calendar\_Entry will specify number of replenishments allowed on a given date. For example, when set to 1, it will only plan one producing operation on each effective schedule date. The default value for calendar entries is 0, see the Calendar\_Entry value NUMBER extension.

Size of each replenishment is computed to maintain at least the 'min\_on\_hand' until the next replenishment observing the lot sizing restrictions.  
Default: [unspecified]

quantity\_range - - a Quantity\_Range field of model  
PRODUCING\_FLOW\_CALENDAR  
The Quantity\_Range between which producing operations of this buffer are constrained to lie.

Note that the smallest legal size of a producing operation can be more than the min of the quantity\_range, since it has to be a multiple of multiple\_quantity. Similarly, the largest legal size can be smaller than the max of the quantity\_range.  
Default: [1, oo]

multiple\_quantity - - a Quantity field of model  
PRODUCING\_FLOW\_CALENDAR  
The Quantity of which the Quantity of the Item to be produced by each 'producing\_operation' that is issued by this PRODUCING\_FLOW\_CALENDAR Buffer is a multiple of.

This Quantity is converted to the unit of this Buffer. If multiple\_quantity is set to "0", it implies that the quantities of producing operations can be any number (integer or non-integer) in the range of "quantity\_range". The default is "0".  
Default: 0

fence . . . a Horizon\_Date field of model PRODUCEING\_FLOW\_CALENDAR  
The horizon-relative Date beyond which this flow\_policy switches to coarser planning estimates. At and after the fence Date 'producing\_operation' issued by this PRODUCEING\_FLOW\_CALENDAR buffer will use 'after\_fence\_quantity\_range' instead of 'quantity\_range' and 'after\_fence\_multiple\_quantity' instead of 'multiple\_quantity' as lot sizing restrictions.

Also, at and after this 'fence', replenishment calendar is not effective causing this flow policy to behave like MULTIPLE. At and after the 'fence' date, it can create producing Operation\_Plans as needed.

This fence is relative to Plan.current.  
Default: 00

after\_fence\_quantity\_range . . . a Quantity\_Range field of model PRODUCEING\_FLOW\_CALENDAR

The range between which producing operations of the buffer are constrained to lie at and after fence.

Note that the smallest legal size of a supplying operation can be more than the min of the quantity\_range, since it has to be a multiple of multiple\_quantity. Similarly, the largest legal size can be smaller than the max of the quantity\_range.  
Default: [1, 00]

after\_fence\_multiple\_quantity . . . a Quantity field of model PRODUCEING\_FLOW\_CALENDAR

The Quantity of the Item to be produced by each 'producing\_operation' that is issued by this Buffer is a multiple of at and after the fence. This Quantity is converted to the unit of this Buffer. An 'after\_fence\_multiple\_quantity' of "0" implies that the quantities of producing operations can be any number in the range of 'quantity\_range'. The default is "0".  
Default: 0

default\_min\_on\_hand . . . a Quantity field of model PRODUCEING\_FLOW\_CALENDAR

The default minimum on\_hand balance that should be present in this Buffer. This will be effective during the periods when there are no Calendar\_Entry's specified in the 'min\_on\_hand' Calendar.

If the 'min\_on\_hand' Calendar is [unspecified], the 'default\_min\_on\_hand' will be effective through out the planning horizon.

If the stocking\_policy is CALENDAR then default\_min\_on\_hand will be automatically computed.  
Default: 0

min\_on\_hand\_calendar . . . a Calendar field of model PRODUCEING\_FLOW\_CALENDAR

A Calendar of entries that define minimum on\_hand balance that should be maintained in this Buffer at those dates. These Quantity's are converted to the unit of this Buffer.

If the stocking\_policy is CALENDAR then min\_on\_hand\_calendar will be automatically computed.  
Default: [unspecified]

default\_excess\_on\_hand . . . a Quantity field of model PRODUCEING\_FLOW\_CALENDAR

Up to this Quantity more than the minimum required is allowed in this Buffer. This is the default excess on\_hand Quantity effective during the periods when there are no Calendar\_Entry's specified in the 'excess\_on\_hand' Calendar.

If the 'excess\_on\_hand' Calendar is [unspecified], the 'default\_excess\_on\_hand' will be effective through out the planning horizon.  
Default: 0

excess\_on\_hand\_calendar . . . a Calendar field of model PRODUCEING\_FLOW\_CALENDAR

A Calendar of entries that define Quantity more than the minimum required is allowed in this Buffer. These Quantity's are converted to the unit of this Buffer.

Setting excess\_on\_hand above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels.  
Default: [unspecified]

default\_min\_time . . . a Time field of model PRODUCEING\_FLOW\_CALENDAR  
This is the default min\_time effective during the periods when there are no Calendar\_Entry's specified in the 'min\_time' Calendar.

Items are planned such that they arrive at least 'default\_min\_time' earlier than otherwise needed. This Buffer will allow specification of date effective min\_time which can vary over time.



If the stocking\_policy is CALENDAR and if the units\_of\_safety\_stock is either TIME or QUANTITY\_TIME then default\_min\_time will be automatically computed.

min\_time is represented internally as seconds so if specified as as days, weeks, years, its will be converted to seconds. This needs to be considered around daylight savings time boundaries.

Default: 0

min\_time\_calendar -- a Calendar field of model

PRODUCING\_FLOW\_CALENDAR

A Calendar of entries that define min\_time effective during those dates. Items are planned such that they arrive at least min\_time earlier than otherwise needed. This Buffer will allow specification of date effective min\_time which can vary over time.

This may be used as a safety time -- the next min\_time of consumption will be maintained in the Buffer.

The min\_time gives a fixed time of consumption that should always be covered (for example, setting min\_time to "3 weeks" will keep enough stock to cover the next 3 weeks of out flows).

If the stocking\_policy is CALENDAR and if the units\_of\_safety\_stock is either TIME or QUANTITY\_TIME then min\_time\_calendar will be automatically computed.

Default: [unspecified]

end\_item -- a Logical field of model PRODUCING\_FLOW\_CALENDAR

If TRUE, this Buffer will assume that majority of it's consuming Operation\_Plans have DELIVER motive and are connected directly to the Item\_Requests/Promises. It will take advantage of this proximity to Requests/Promises and resolve problems by selecting Operation\_Plans and actions(MOVE\_IN, MOVE\_OUT, RESIZE\_MORE, RESIZE\_LESS etc.), which will result into reduced lateness, shortness or in general most positive impact on the Active\_Goals.

If FALSE, this Buffer will use internal heuristics to select Operation\_Plans and actions(MOVE\_IN, MOVE\_OUT, RESIZE\_MORE, RESIZE\_LESS etc.) to solve Problems.

Default: false

rank\_delivery\_operation\_plan\_higher -- a Logical field of model

PRODUCING\_FLOW\_CALENDAR

If TRUE, consuming Operation\_Plans motivated directly by Requests and Promises via DELIVER motive are ranked higher than the consuming Operation\_Plans motivated by downstream Buffers or Resources.

If FALSE, consuming Operation\_Plans motivated directly by Requests and Promises via DELIVER motive are ranked lower. This will be used by problem resolvers to rank candidates while taking Strategy allowed actions such as MOVE\_OUT, RESIZE\_LESS.

Default: true

producing\_operation -- a Operation field of model

PRODUCING\_FLOW\_CALENDAR

The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field\_Error for this field to be [unspecified], the default, if any Buffer\_Plan needs to issue a producing Operation\_Plan.

Default: [unspecified]

PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK -- a flow\_policy extension of model Buffer

Just like Producing Flow Policy, but with user entry points to control resolver behavior

The PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK model has fields that references these models :

Calendar, Operation.

calendar -- a Calendar field of model

PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK

A Calendar of entries that define the preset schedule of replenishments. Each Calendar\_Entry will specify number of replenishments allowed on a given date. Defaulted to 0, it will not plan producing operation on each effective schedule date unless the size is set.

Size of each replenishment is computed to maintain at least the min\_on\_hand until the next replenishment observing the lot sizing restrictions.

Default: [unspecified]

quantity\_range - - *a Quantity\_Range field of model*

**PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

The Quantity\_Range between which producing operations of this buffer are constrained to lie.

Note that the smallest legal size of a producing operation can be more than the min of the quantity\_range, since it has to be a multiple of multiple\_quantity. Similarly, the largest legal size can be smaller than the max of the quantity\_range.

Default: [1, 00]

multiple\_quantity - - *a Quantity field of model*

**PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

The Quantity of which the Quantity of the Item to be produced by each producing\_operation that is issued by this PRODUCING\_FLOW\_CALENDAR Buffer is a multiple of.

This Quantity is converted to the unit of this Buffer. The default is "1", which means 1 unit of the Buffer.

Default: 1

fence - - *a Horizon\_Date field of model*

**PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

The horizon-relative Date beyond which this flow\_policy switches to coarser planning estimates. At and after the fence Date 'producing\_operation' issued by this PRODUCING\_FLOW\_CALENDAR buffer will use 'after\_fence\_quantity\_range' instead of 'quantity\_range' and 'after\_fence\_multiple\_quantity' instead of 'multiple\_quantity' as lot sizing restrictions.

Also, at and after this 'fence', replenishment calendar is not effective causing this flow policy to behave like MULTIPLE. After the 'fence' date, it can create producing Operation\_Plans as needed.

This fence is relative to Plan.current.

Default: 00

after\_fence\_quantity\_range - - *a Quantity\_Range field of model*

**PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

The range between which producing operations of the buffer are constrained to lie at and after fence.

Note that the smallest legal size of a supplying operation can be more than the min of the quantity\_range, since it has to be a multiple of multiple\_quantity. Similarly, the largest legal size can be smaller than the max of the quantity\_range.

Default: [1, 00]

after\_fence\_multiple\_quantity - - *a Quantity field of model*

**PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

The Quantity of the Item to be produced by each 'producing\_operation' that is issued by this Buffer is a multiple of at and after the fence. This Quantity is converted to the unit of this Buffer. The default is "1", which means 1 unit of the Buffer.

Default: 1

default\_min\_on\_hand - - *a Quantity field of model*

**PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

The default minimum on\_hand balance that should be present in this Buffer. This will be effective during the periods when there are no Calendar\_Entry's specified in the min\_on\_hand' Calendar.

If the 'min\_on\_hand' Calendar is [unspecified], the 'default\_min\_on\_hand' will be effective through out the planning horizon.

Default: 0

min\_on\_hand\_calendar - - *a Calendar field of model*

**PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

A Calendar of entries that define minimum on\_hand balance that should be maintained in this Buffer at those dates. These Quantity's are converted to the unit of this Buffer.

Default: [unspecified]

default\_excess\_on\_hand - - *a Quantity field of model*

**PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

Up to this Quantity more than the minimum required is allowed in this Buffer. This is the default excess on\_hand Quantity effective during the periods when there are no Calendar\_Entry's specified in the 'excess\_on\_hand' Calendar.

If the 'excess\_on\_hand' Calendar is [unspecified], the 'default\_excess\_on\_hand' will be effective through out the planning horizon.

Default: 0

excess\_on\_hand\_calendar -- *a Calendar field of model*

**PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

A Calendar of entries that define Quantity more than the minimum required is allowed in this Buffer. These Quantity's are converted to the unit of this Buffer.

Setting excess\_on\_hand above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels.

Default: [unspecified]

default\_min\_time -- *a Time field of model*

**PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

This is the default min\_time effective during the periods when there are no Calendar\_Entry's specified in the min\_time Calendar.

Items are planned such that they arrive at least 'default\_min\_time' earlier than otherwise needed. This Buffer will allow specification of date effective min\_time which can vary over time.

'min\_time' is represented internally as seconds so if specified as as days, weeks, years, it will be converted to seconds. This needs to be considered around daylight savings time boundaries.

Default: 0

min\_time\_calendar -- *a Calendar field of model*

**PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

A Calendar of entries that define min\_time effective during those dates. Items are planned such that they arrive at least 'min\_time' earlier than otherwise needed. This Buffer will allow specification of date effective min\_time which can vary over time.

This may be used as a safety time -- the next min\_time of consumption will be maintained in the Buffer.

The 'min\_time' gives a fixed time of consumption that should always be covered (for example, setting 'min\_time' to "3 weeks" will keep enough stock to cover the next 3 weeks of out flows).

Default: [unspecified]

end\_item -- *a Logical field of model*

**PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

If TRUE, this Buffer will assume that majority of it's consuming Operation\_Plans have DELIVER motive and are connected directly to the Item\_Requests/Promises. It will take advantage of this proximity to Requests/Promises and resolve problems by selecting Operation\_Plans and actions(MOVE\_IN, MOVE\_OUT, RESIZE\_MORE, RESIZE\_LESS etc.), which will result into reduced lateness, shortness or in general most positive impact on the Active\_Goals.

If FALSE, this Buffer will use internal heuristics to select Operation\_Plans and actions(MOVE\_IN, MOVE\_OUT, RESIZE\_MORE, RESIZE\_LESS etc.) to solve Problems.

Default: false

rank\_delivery\_operation\_plan\_higher -- *a Logical field of model*

**PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

If TRUE, consuming Operation\_Plans motivated directly by Requests and Promises via DELIVER motive are ranked higher than the consuming Operation\_Plans motivated by downstream Buffers or Resources.

If FALSE, consuming Operation\_Plans motivated directly by Requests and Promises via DELIVER motive are ranked lower. This will be used by problem solvers to rank candidates while taking Strategy allowed actions such as MOVE\_OUT, RESIZE\_LESS.

Default: true

producing\_operation -- *a Operation field of model*

**PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field\_Error for this field to be [unspecified], the default, if any Buffer\_Plan needs to issue a producing Operation\_Plan.

Default: [unspecified]

flow\_plan\_selection\_start -- *a Expression field of model*

**PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

This expression is called before Flow\_Plans are selected in order to build bags. For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem active\_strategy : Bound to Active\_Strategy solving problem on this Buffer.

Example Usage:  
Default: [unspecified]

flow\_plan\_selection\_end - - *a Expression field of model*  
PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK

This expression is called after Flow\_Plans are selected to build the bags. For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem active\_strategy : Bound to Active\_Strategy solving problem on this Buffer.

Example Usage:  
Default: [unspecified]

flow\_plan\_selection\_interval - - *a Expression field of model*  
PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK

This expression returns the Date\_Range over which resolver will look to select Flow\_Plans in order to build bags. For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer.

Example Usage:  
Buffer b1 = supply\_chains.find("Sup1").sites.find("Site1").buffers.find(Bu1)  
b1.set\_flow\_plan\_selection\_interval("Date\_Range(problem\_start - 1 Week, problem\_end)")  
Default: problem\_period

flow\_plan\_filter - - *a Expression field of model*  
PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK

This expression takes following parameters and if it returns true, the Flow\_Plan will be selected and put into the bag. If it returns false, the Flow\_Plan will not be put into the bag.

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: true

flow\_plan\_rank - - *a Expression field of model*  
PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK

This expression returns the rank of the Flow\_Plan which is used by the resolver in selecting the Flow\_Plan. Higher rank means that the Flow\_Plan is more likely to be chosen later from the bag.

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: 1.0

continue\_flow\_plan\_selection - - *a Expression field of model*  
PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK

This expression takes following parameters and if it returns true, the selection of Flow\_Plan will continue to be put into the bag. If it returns false, further Flow\_Plans will not be selected.

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: true

flow\_plan\_resize\_quantity\_range - - *a Expression field of model*  
PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK  
Replan restrictions This expression returns the Quantity\_Range within which the selected Flow\_Plan can be resized by the resolver.

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: [unspecified]

flow\_plan\_move\_restriction - - *a Expression field of model*  
PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK  
This expression returns the Date\_Range within which the selected Flow\_Plan can be moved by the resolver.

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: [unspecified]

flow\_plan\_split\_restriction - - *a Expression field of model*  
PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK  
This expression returns the Quantity\_Range within which the selected Flow\_Plan can be split by the resolver for a move

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: [unspecified]

SUPPLY\_CALENDAR -- *a flow\_policy extension of model* Buffer

A SUPPLY\_CALENDAR Buffer is replenished artificially as specified in a Calendar. The 'calendar' specifies additional supply "flows" producing into the Buffer. This can be used as a simple way to model regular deliveries from a reliable supplier. No Operations are generated; no capacity or items are consumed; and no purchase Requests. The buffer is just replenished "spontaneously", on its own.

It may also be used to model other requirements that are easily modeled as an additional material requirement. For example, the number of starts into a facility may be limited to a certain number each month. The easiest way to model that may be as an additional Item that is needed by the first Operation. The quantity of that Item available each month limits the number of starts. Note that unused starts from the previous months will carry over. See the ON\_HAND\_CALENDAR flow\_policy for the same that does not carry over.

The SUPPLY\_CALENDAR model has fields that references these models :  
Calendar.

calendar - - a Calendar field of model SUPPLY\_CALENDAR

A Calendar of QUANTITY entries that each specify a supply producing into the Buffer. Unconsumed supply accumulates in the Buffer. Note that any default\_quantity set for this calendar will be ignored, since its impact on on\_hand is impossible to determine without a day\_pattern.  
Default: [unspecified]

ON\_HAND\_CALENDAR -- a flow\_policy extension of model Buffer

A ON\_HAND\_CALENDAR Buffer is replenished artificially as specified in a Calendar. The 'calendar' specifies the new on\_hand balance at that Date (as if whatever flow is necessary to give that balance will arrive or depart at that Date). This can be used as a simple way to model regular deliveries from a reliable supplier. No Operations are generated; no capacity or Items are consumed; and no purchase Requests. The buffer is just replenished "spontaneously", on its own.

It may also be used to model other requirements that are easily modeled as an additional material requirement. For example, the number of starts into a facility may be limited to a certain number each month. The easiest way to model that may be as an additional Item that is needed by the first Operation. The quantity of that Item available each month limits the number of starts. Note that unused starts from the previous months will not carry over. See the SUPPLY\_CALENDAR flow\_policy for the same that does carry over.

The ON\_HAND\_CALENDAR model has fields that references these models :  
Calendar.

calendar - - a Calendar field of model ON\_HAND\_CALENDAR  
A Calendar of QUANTITY entries that each specify a new on-hand balance for the Buffer. Production or consumption occurs spontaneously such that the specified balance occurs. Note that any default\_quantity set for this calendar will be ignored, since its impact on on\_hand is impossible to determine without a day\_pattern.  
Default: [unspecified]

ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK -- a flow\_policy extension of model Buffer

A ON\_HAND\_CALENDAR Buffer is replenished artificially as specified in a Calendar. The 'calendar' specifies the new on\_hand balance at that Date (as if whatever flow is necessary to give that balance will arrive or depart at that Date). This can be used as a simple way to model regular deliveries from a reliable supplier. No Operations are generated; no capacity or Items are consumed; and no purchase Requests. The buffer is just replenished "spontaneously", on its own.

It may also be used to model other requirements that are easily modeled as an additional material requirement. For example, the number of starts into a facility may be limited to a certain number each month. The easiest way to model that may be as an additional Item that is needed by the first Operation. The quantity of that Item available each month limits the number of starts. Note that unused starts from the previous months will not carry over. See the SUPPLY\_CALENDAR flow\_policy for the same that does carry over.

The ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK model has fields that references these models :  
Calendar.

calendar - - a Calendar field of model

ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK

A Calendar of QUANTITY entries that each specify a new on-hand balance for the Buffer. Production or consumption occurs spontaneously such that the specified balance occurs. Note that any default\_quantity set for this calendar will be ignored, since its impact on on\_hand is impossible to determine without a day\_pattern.  
Default: [unspecified]

flow\_plan\_selection\_start - - a Expression field of model

ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK

This expression is called before Flow\_Plans are selected in order to build bags. For the convenience of OIL expression writer following special variables are available.

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem active\_strategy : Bound to Active\_Strategy solving problem on this Buffer.

Example Usage:  
Default: [unspecified]

flow\_plan\_selection\_end - - a Expression field of model  
ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK

This expression is called after Flow\_Plans are selected to build the bags. For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem active\_strategy : Bound to Active\_Strategy solving problem on this Buffer.

Example Usage:  
Default: [unspecified]

flow\_plan\_selection\_interval - - a Expression field of model  
ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK

This expression returns the Date\_Range over which resolver will look to select Flow\_Plans in order to build bags. For the convenience of OIL expression writer following special variables are available.

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer.

Example Usage:  
Buffer b1 = supply\_chains.find("Sup1").sites.find("Site1").buffers.find(Buf1)  
b1.set\_flow\_plan\_selection\_interval("Date\_Range(problem\_start - 1 Week, problem\_end)")  
Default: problem\_period

flow\_plan\_filter - - a Expression field of model  
ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK  
This expression takes following parameters and if it returns true, the Flow\_Plan will be selected and put into the bag. If it returns false, the Flow\_Plan will not be put into the bag.

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: true

flow\_plan\_rank - - a Expression field of model  
ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK

This expression returns the 'rank' of the Flow\_Plan which is used by the resolver in selecting the Flow\_Plan. Higher 'rank' means that the Flow\_Plan is more likely to be chosen later from the bag.

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: 1.0

continue\_flow\_plan\_selection - - *a* Expression field of model  
ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK  
This expression takes following parameters and if it returns true, the selection of Flow\_Plan will continue to be put into the bag. If it returns false, further Flow\_Plans will not be selected.

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer.problem\_period : Bound to the problem period of the currently selected problem.problem\_category : Bound to Problem\_Category of the currently selected problem.change\_category : Bound to Strategy\_Change.category for which resolver is building bag.active\_strategy : Bound to Active\_Strategy solving problem on this Buffer.flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: true

flow\_plan\_resize\_quantity\_range - - *a* Expression field of model  
ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK  
Replan restrictions This expression returns the Quantity\_Range within which the selected Flow\_Plan can be resized by the resolver.

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer.problem\_period : Bound to the problem period of the currently selected problem.problem\_category : Bound to Problem\_Category of the currently selected problem.change\_category : Bound to Strategy\_Change.category for which resolver is building bag.active\_strategy : Bound to Active\_Strategy solving problem on this Buffer.flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: [unspecified]

flow\_plan\_move\_restriction - - *a* Expression field of model  
ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK  
This expression returns the Date\_Range within which the selected Flow\_Plan can be moved by the resolver. For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer.problem\_period : Bound to the problem period of the currently selected problem.problem\_category : Bound to Problem\_Category of the currently selected problem.change\_category : Bound to Strategy\_Change.category for which resolver is building bag.active\_strategy : Bound to Active\_Strategy solving problem on this Buffer.flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: [unspecified]

flow\_plan\_split\_restriction - - *a* Expression field of model  
ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK  
This expression returns the Quantity\_Range within which the selected Flow\_Plan can be split by the resolver for a move

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer.problem\_period : Bound to the problem period of the currently selected problem.problem\_category : Bound to Problem\_Category of the currently selected problem.change\_category : Bound to Strategy\_Change.category for which resolver is building bag.active\_strategy : Bound to Active\_Strategy solving problem on this Buffer.flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: [unspecified]

FLOW\_LIMIT\_CALENDAR -- *a* flow\_policy extension of model Buffer

A FLOW\_LIMIT\_CALENDAR Buffer has a behaviour identical to a ON\_HAND\_CALENDAR except that the NEGATIVE\_ON\_HAND problems are in this case called OVER\_FLOW\_LIMIT problems. This can be used to model unit capacity buffers. See ON\_HAND\_CALENDAR for more details on how this flow policy is supposed to work.

The FLOW\_LIMIT\_CALENDAR model has fields that references these models :  
Calendar.



**calendar** - - *a Calendar field of model FLOW\_LIMIT\_CALENDAR*  
A Calendar of QUANTITY entries that each specify a new on-hand balance for the Buffer. Supply or consumption occurs spontaneously such that the specified balance occurs.  
Default: [unspecified]

**FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK** - - *a flow\_policy extension of model Buffer*

A FLOW\_LIMIT\_CALENDAR Buffer has a behaviour identical to a ON\_HAND\_CALENDAR except that the NEGATIVE\_ON\_HAND problems are in this case called OVER\_FLOW\_LIMIT problems. This can be used to model unit capacity buffers. See ON\_HAND\_CALENDAR for more details on how this flow policy is supposed to work.

The FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK model has fields that references these models :  
Calendar:

**calendar** - - *a Calendar field of model*  
**FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK**  
A Calendar of QUANTITY entries that each specify a new on-hand balance for the Buffer. Supply or consumption occurs spontaneously such that the specified balance occurs.  
Default: [unspecified]

**flow\_plan\_selection\_start** - - *a Expression field of model*  
**FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK**  
This expression is called before Flow\_Plan are selected in order to build bags. For the convenience of OIL expression writer following special variables are available,

**buffer\_plan** : Bound to Buffer\_Plan of this Buffer. **problem\_period** : Bound to the problem period of the currently selected problem. **problem\_category** : Bound to Problem\_Category of the currently selected problem **active\_strategy** : Bound to Active\_Strategy solving problem on this Buffer.

Example Usage:  
Default: [unspecified]

**flow\_plan\_selection\_end** - - *a Expression field of model*  
**FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK**  
This expression is called after Flow\_Plan are selected to build the bags. For the convenience of OIL expression writer following special variables are available,

**buffer\_plan** : Bound to Buffer\_Plan of this Buffer. **problem\_period** : Bound to the problem period of the currently selected problem. **problem\_category** : Bound to Problem\_Category of the currently selected problem **active\_strategy** : Bound to Active\_Strategy solving problem on this Buffer.

Example Usage:  
Default: [unspecified]

**flow\_plan\_selection\_interval** - - *a Expression field of model*  
**FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK**  
This expression returns the Date\_Range over which resolver will look to select Flow\_Plan in order to build bags. For the convenience of OIL expression writer following special variables are available,

**buffer\_plan** : Bound to Buffer\_Plan of this Buffer. **problem\_period** : Bound to the problem period of the currently selected problem. **problem\_category** : Bound to Problem\_Category of the currently selected problem **change\_category** : Bound to Strategy\_Change.category for which resolver is building bag. **active\_strategy** : Bound to Active\_Strategy solving problem on this Buffer.

Example Usage:  
**Buffer** b1 = supply\_chains.find("Sup1").sites.find("Site1").buffers.find(Buffer1)  
**b1.set\_flow\_plan\_selection\_interval("Date\_Range(problem\_start - 1 Week, problem\_end)")**  
Default: **problem\_period**

**flow\_plan\_filter** - - *a Expression field of model*  
**FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK**  
This expression takes following parameters and if it returns true, the Flow\_Plan will be selected and put into the bag. If it returns false, the Flow\_Plan will not be put into the bag.

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: true

flow\_plan\_rank -- a Expression field of model  
FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK

This expression returns the 'rank' of the Flow\_Plan which is used by the resolver in selecting the Flow\_Plan. Higer 'rank' means that the Flow\_Plan is more likely to be chosen later from the bag.

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: 1.0

continue\_flow\_plan\_selection -- a Expression field of model  
FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK

This expression takes following parameters and if it returns true, the selection of Flow\_Plan will continue to be put into the bag. If it returns false, further Flow\_Plans will not be selected.

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: true

flow\_plan\_resize\_quantity\_range -- a Expression field of model  
FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK

Replan restrictions This expression returns the Quantity\_Range within which the selected Flow\_Plan can be resized by the resolver.

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: [unspecified]

flow\_plan\_move\_restriction -- a Expression field of model  
FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK

This expression returns the Date\_Range within which the selected Flow\_Plan can be moved by the resolver. For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:

Default: [unspecified]

**flow\_plan\_split\_restriction -- a Expression field of model FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK**

This expression returns the Quantity\_Range within which the selected Flow\_Plan can be split by the resolver for a move

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer.  
problem\_period: : Bound to the problem period of the currently selected problem.  
problem\_category: : Bound to Problem\_Category of the currently selected problem.  
change\_category: : Bound to Strategy\_Change\_category for which resolver is building bag.  
active\_strategy: : Bound to Active\_Strategy solving problem on this Buffer.  
flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: [unspecified]

**INFINITE -- a flow\_policy extension of model Buffer**

An INFINITE Buffer models the flows that are produced into or consumed from it, but detects no Problems and will never create Operations to replenish itself. It allows you to display the flow of material, without imposing any constraints on the plan due to that flow.

This is often used for raw materials where the purchasing of those materials is not planned by this system. For example, an inventory that is effectively managed via reorder-point can be modelled in this system as an INFINITE Buffer.

Note that this can model not only the infinite supply of an abundant material, but also the infinite disposal or consumption of an uninteresting material.

Problems Identified and Resolved by INFINITE buffer:

INFINITE flow policy doesnot identify any problems. The only time an INFINITE buffer will identify problems is when user has attached Buffer\_Problem. Detectors to the buffer. These problem detectors will identify problems regardless of flow policy.

**SUPPLIER -- a flow\_policy extension of model Buffer**

A SUPPLIER Buffer models flows into the system from an external (not modelled) source. Requests which originate from downstream are satisfied if the material requirement date is at or after the supplier fence, and not satisfied if the request is nearer the current date. In that sense, this is like an INFINITE flow policy, with the additional restriction that it is only INFINITE at and after the fence.

This is used to model orders from suppliers with restrictions on when an order can be processed.

This is often used for raw materials where the purchasing of those materials is not planned by this system. For example, an inventory that is effectively managed via reorder-point but requires advance purchase planning can be modelled in this system as a SUPPLIER Buffer.

fence - - a Horizon\_Date field of model SUPPLIER  
At and after the fence, this flow policy behaves exactly like an INFINITE flow policy. Currently, inside the fence, no requests are satisfied.  
Default: oo

**BASIC -- a flow\_policy extension of model Buffer**

A BASIC Buffer maintains a 'min\_time' safety time and a 'min\_on\_hand' safety quantity. The 'min\_on\_hand' gives a fixed minimum level for the safety stock. The 'min\_time' gives a fixed time of consumption that should always be covered (for example, setting 'min\_time' to "3 weeks" will keep enough stock to cover the next 3 weeks of out flows).

Up to 'excess\_on\_hand' more than the minimum required is allowed in the Buffer. By setting 'excess\_on\_hand' larger than zero, planning instability is reduced, and planning speed is increased. However, keeping it smaller minimizes excess inventory.

For example, consider a Buffer with 'min\_on\_hand' set to "100", 'min\_time' set to "2 weeks", and 'excess\_on\_hand' set to "20". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the on\_hand maintained until 4/1 will be "100", the 'min\_on\_hand'. From 4/1 to 4/15, "200" will be maintained in order to cover the next "2 weeks" of out flows. If the out flow is reduced to "190", the producing operation plans may not change -- since "200" is only "10" greater than "190", and "10" is less than "20", the 'excess\_on\_hand' that is allowed.

The BASIC model has fields that references these models :  
Operation.

Extensions	BASIC_FILTER_AND_RANK Extension
------------	---------------------------------

**min\_time** -- *a Time field of model BASIC*  
Items are planned such that they arrive at least 'min\_time' earlier than otherwise needed. This may be used as a safety time -- the next 'min\_time' of consumption will be maintained in the Buffer. 'min\_time' is represented internally as seconds so if specified as as days, weeks, years, its will be converted to seconds. This needs to be considered around daylight savings time boundaries.  
Default: 0

**min\_on\_hand** -- *a Quantity field of model BASIC*  
The minimum on\_hand balance that should be present in this Buffer. This Quantity is converted to the 'unit' of this Buffer.  
Default: 0

**excess\_on\_hand** -- *a Quantity field of model BASIC*  
Up to this Quantity more than the minimum required is allowed in this Buffer. Setting this above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels. This Quantity is converted to the 'unit' of this Buffer.  
Default: 0

**producing\_operation** -- *a Operation field of model BASIC*  
The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field\_Error for this field to be [unspecified], the default, if any Buffer\_Plan needs to issue a producing Operation\_Plan.  
Default: [unspecified]

**supplying\_operation** -- *a Operation field of model BASIC*  
Obsolete! This field has been renamed 'producing\_operation' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.  
Default: [unspecified]  
Properties: obsolete=True

**BASIC\_FILTER\_AND\_RANK** -- *a flow\_policy extension of model Buffer*

Just like Basic Flow Policy, but with user entry points to control resolver behavior

The BASIC\_FILTER\_AND\_RANK model has fields that references these models :  
Operation.

Extensions	BASIC_FILTER_AND_RANK Extension
------------	---------------------------------

**min\_time** -- *a Time field of model BASIC\_FILTER\_AND\_RANK*  
Items are planned such that they arrive at least 'min\_time' earlier than otherwise needed. This may be used as a safety time -- the next 'min\_time' of consumption will be maintained in the Buffer. 'min\_time' is represented internally as seconds so if specified as as days, weeks, years, its will be converted to seconds. This needs to be considered around daylight savings time boundaries.  
Default: 0

**min\_on\_hand** -- *a Quantity field of model BASIC\_FILTER\_AND\_RANK*  
The minimum on\_hand balance that should be present in this Buffer. This Quantity is converted to the 'unit' of this Buffer.  
Default: 0

**excess\_on\_hand** -- *a Quantity field of model BASIC\_FILTER\_AND\_RANK*  
Up to this Quantity more than the minimum required is allowed in this Buffer. Setting this above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels. This Quantity is converted to the 'unit' of this Buffer.  
Default: 0

**producing\_operation** -- *a Operation field of model BASIC\_FILTER\_AND\_RANK*  
The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field\_Error for this field to be [unspecified], the default, if any Buffer\_Plan needs to issue a producing Operation\_Plan.  
Default: [unspecified]

**supplying\_operation** -- *a Operation field of model BASIC\_FILTER\_AND\_RANK*  
Obsolete! This field has been renamed 'producing\_operation' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.  
Default: [unspecified]  
Properties: obsolete=True

**flow\_plan\_selection\_start** -- *a Expression field of model BASIC\_FILTER\_AND\_RANK*  
This expression is called before Flow\_Plans are selected in order to build bags. For the convenience of OIL expression writer following special variables are available.

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem active\_strategy : Bound to Active\_Strategy solving problem on this Buffer.

Example Usage:  
Default: [unspecified]

flow\_plan\_selection\_end - - *a Expression field of model*  
BASIC\_FILTER\_AND\_RANK

This expression is called after Flow\_Plans are selected to build the bags. For the convenience of OIL expression writer following special variables are available.

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem active\_strategy : Bound to Active\_Strategy solving problem on this Buffer.

Example Usage:  
Default: [unspecified]

flow\_plan\_selection\_interval - - *a Expression field of model*  
BASIC\_FILTER\_AND\_RANK

This expression returns the Date\_Range over which resolver will look to select Flow\_Plans in order to build bags. For the convenience of OIL expression writer following special variables are available.

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer.

Example Usage:

```
Buffer b1 = supply_chains.find("Sup1").sites.find("Site1").buffers.find(Buffer)
b1.sel_flow_plan_selection_interval("Date_Range(problem_start - 1 Week,
problem_end)")
Default: problem_period
```

flow\_plan\_filter - - *a Expression field of model* BASIC\_FILTER\_AND\_RANK  
This expression takes following parameters and if it returns true, the Flow\_Plan will be selected and put into the bag. If it returns false, the Flow\_Plan will not be put into the bag.

For the convenience of OIL expression writer following special variables are available.

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: true

flow\_plan\_rank - - *a Expression field of model* BASIC\_FILTER\_AND\_RANK  
This expression returns the 'rank' of the Flow\_Plan which is used by the resolver in selecting the Flow\_Plan. Higher 'rank' means that the Flow\_Plan is more likely to be chosen later from the bag.

For the convenience of OIL expression writer following special variables are available.

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: 1.0

```
continue_flow_plan_selection - - a Expression field of model
BASIC_FILTER_AND_RANK
This expression takes following parameters and if it returns true, the selection of
Flow_Plan will continue to be put into the bag. If it returns false, further Flow_Plans
will not be selected.
```

For the convenience of OIL expression writer following special variables are available.

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: true

flow\_plan\_resize\_quantity\_range - - *a Expression field of model*  
BASIC\_FILTER\_AND\_RANK  
Replan restrictions This expression returns the Quantity\_Range within which the selected Flow\_Plan can be resized by the resolver.

For the convenience of OIL expression writer following special variables are available.

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: [unspecified]

flow\_plan\_move\_restriction - - *a Expression field of model*  
BASIC\_FILTER\_AND\_RANK

This expression returns the Date\_Range within which the selected Flow\_Plan can be moved by the resolver.

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: [unspecified]

flow\_plan\_split\_restriction - - *a Expression field of model*  
BASIC\_FILTER\_AND\_RANK  
This expression returns the Quantity\_Range within which the selected Flow\_Plan can be split by the resolver for a move

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: [unspecified]

FIXED\_QUANTITY - - *a flow\_policy extension of model* Buffer

A FIXED\_QUANTITY Buffer issues replenishment Operations all with the same 'quantity'. The Operations are planned to arrive such that the Buffer's on\_hand balance never drops below 'min\_on\_hand'. This policy is often called "Fixed Order Quantity (FOQ)".

Up to 'excess\_on\_hand' more than the minimum required is allowed in the Buffer. By setting 'excess\_on\_hand' larger than zero, planning instability is reduced, and planning speed is increased. However, keeping it smaller minimizes excess inventory.

The FIXED\_QUANTITY model has fields that references these models :  
Operation.

Extensions	FIXED_QUANTITY Extension
------------	--------------------------

**min\_on\_hand** - - *a Quantity field of model FIXED\_QUANTITY*  
The minimum on\_hand balance that should be present in this Buffer. This Quantity is converted to the 'unit' of this Buffer.  
Default: 0

**min\_time** - - *a Time field of model FIXED\_QUANTITY*  
Items are planned such that they arrive at least 'min\_time' earlier than otherwise needed. This may be used as a safety time -- the next 'min\_time' of consumption will be maintained in the Buffer. 'min\_time' is represented internally as seconds so if specified as days, weeks, years, its will be converted to seconds. This needs to be considered around daylight savings time boundaries.  
Default: 0

**excess\_on\_hand** - - *a Quantity field of model FIXED\_QUANTITY*  
Up to this Quantity more than the minimum required is allowed in this Buffer. Setting this above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels. This Quantity is converted to the 'unit' of this Buffer.  
Default: 0

**quantity** - - *a Quantity field of model FIXED\_QUANTITY*  
The fixed Quantity of the Item to be produced by each 'producing\_operation' that is issued by this FIXED\_QUANTITY Buffer. This Quantity is converted to the 'preferred\_measure' of this Buffer. The default is "1", which means 1 unit of the Buffer.  
Default: 1

**producing\_operation** - - *a Operation field of model FIXED\_QUANTITY*  
The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.  
It is a Field\_Error for this field to be [unspecified], the default, if any Buffer\_Plan needs to issue a producing Operation\_Plan.  
Default: [unspecified]

**supplying\_operation** - - *a Operation field of model FIXED\_QUANTITY*  
Obsolete! This field has been renamed 'producing\_operation' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.  
Default: [unspecified]  
Properties: obsolete=True

Extensions	MULTIPLE Extension
------------	--------------------

### MULTIPLE - - *a flow\_policy extension of model Buffer*

A MULTIPLE Buffer issues replenishment Operations whose sizes are integral multiples of multiple\_quantity and are bounded by quantity\_range. The Operations are planned to arrive such that the Buffer's on\_hand balance never drops below 'min\_on\_hand'.

Up to 'excess\_on\_hand' more than the minimum required is allowed in the Buffer. By setting 'excess\_on\_hand' larger than zero, planning instability is reduced, and planning speed is increased. However, keeping it smaller minimizes excess inventory.

The MULTIPLE model has fields that references these models :  
**Operation.**

**min\_on\_hand** - - *a Quantity field of model MULTIPLE*  
The minimum on\_hand balance that should be present in this Buffer. This Quantity is converted to the 'unit' of this Buffer.  
Default: 0

**excess\_on\_hand** - - *a Quantity field of model MULTIPLE*  
Up to this Quantity more than the minimum required is allowed in this Buffer. Setting this above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels. This Quantity is converted to the 'unit' of this Buffer.  
Default: 0

**quantity\_range** - - *a Quantity\_Range field of model MULTIPLE*  
The range between which supplying operations of the buffer are constrained to lie.

Note that the smallest legal size of a supplying operation can be more than the min of the quantity\_range, since it has to be a multiple of multiple\_quantity. Similarly, the largest legal size can be smaller than the max of the quantity\_range.  
Default: [1, oo]

**multiple\_quantity** - - *a Quantity field of model MULTIPLE*  
The Quantity of which the Quantity of the Item to be produced by each 'producing\_operation' that is issued by this MULTIPLE Buffer is a multiple of. This Quantity is converted to the unit of this Buffer. If multiple\_quantity is set to "0", it implies that the quantities of producing operations can be any number (integer or non-integer) in the range of "quantity\_range". The default is "0".  
Default: 0

**min\_time** - - *a Time field of model MULTIPLE*

Items are planned such that they arrive at least 'min\_time' earlier than otherwise needed. This may be used as a safety time -- the next 'min\_time' of consumption will be maintained in the Buffer. 'min\_time' is represented internally as seconds so if specified as as days, weeks, years, its will be converted to seconds. This needs to be considered around daylight savings time boundaries.

Default: 0

**producing\_operation** - - *a Operation field of model MULTIPLE*

The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field\_Error for this field to be [unspecified], the default, if any Buffer\_Plan needs to issue a producing Operation\_Plan.

Default: [unspecified]

**supplying\_operation** - - *a Operation field of model MULTIPLE*

Obsolete! This field has been renamed 'producing\_operation' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Default: [unspecified]

Properties: obsolete=True

**MULTIPLE\_FILTER\_AND\_RANK** - - *a flow\_policy extension of model Buffer*

A MULTIPLE Buffer issues replenishment Operations whose sizes are integral multiples of multiple\_quantity and are bounded by quantity\_range. The Operations are planned to arrive such that the Buffer's on\_hand balance never drops below min\_on\_hand.

Up to 'excess\_on\_hand' more than the minimum required is allowed in the Buffer. By setting 'excess\_on\_hand' larger than zero, planning instability is reduced, and planning speed is increased. However, keeping it smaller minimizes excess inventory.

The MULTIPLE\_FILTER\_AND\_RANK model has fields that references these models :  
Operation.

**min\_on\_hand** - - *a Quantity field of model MULTIPLE\_FILTER\_AND\_RANK*

The minimum on\_hand balance that should be present in this Buffer. This Quantity is converted to the unit of this Buffer.

Default: 0

**excess\_on\_hand** - - *a Quantity field of model MULTIPLE\_FILTER\_AND\_RANK*

Up to this Quantity more than the minimum required is allowed in this Buffer. Setting this above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels. This Quantity is converted to the unit of this Buffer.

Default: 0

**quantity\_range** - - *a Quantity\_Range field of model*

**MULTIPLE\_FILTER\_AND\_RANK**

The range between which supplying operations of the buffer are constrained to lie.

Note that the smallest legal size of a supplying operation can be more than the min of the quantity\_range, since it has to be a multiple of multiple\_quantity. Similarly, the largest legal size can be smaller than the max of the quantity\_range.

Default: [1, 00]

**multiple\_quantity** - - *a Quantity field of model*

**MULTIPLE\_FILTER\_AND\_RANK**

The Quantity of which the Quantity of the Item to be produced by each 'producing\_operation' that is issued by this MULTIPLE Buffer is a multiple of. This Quantity is converted to the unit of this Buffer. If multiple\_quantity is set to "0", it implies that the quantities of producing operations can be any number (integer or non-integer) in the range of "quantity\_range". The default is "0".

Default: 0

**min\_time** - - *a Time field of model MULTIPLE\_FILTER\_AND\_RANK*

Items are planned such that they arrive at least 'min\_time' earlier than otherwise needed. This may be used as a safety time -- the next 'min\_time' of consumption will be maintained in the Buffer. 'min\_time' is represented internally as seconds so if specified as as days, weeks, years, its will be converted to seconds. This needs to be considered around daylight savings time boundaries.

Default: 0

**producing\_operation** - - *a Operation field of model*

**MULTIPLE\_FILTER\_AND\_RANK**

The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field\_Error for this field to be [unspecified], the default, if any Buffer\_Plan needs to issue a producing Operation\_Plan.

Default: [unspecified]



supplying\_operation - - *a Expression field of model*  
MULTIPLE\_FILTER\_AND\_RANK

Obsolete! This field has been renamed 'producing\_operation' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Default: [unspecified]  
Properties: obsolete=True

flow\_plan\_selection\_start - - *a Expression field of model*  
MULTIPLE\_FILTER\_AND\_RANK

This expression is called before Flow\_Plan are selected in order to build bags. For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer.problem\_period : Bound to the problem period of the currently selected problem.problem\_category : Bound to Problem\_Category of the currently selected problem.active\_strategy : Bound to Active\_Strategy solving problem on this Buffer.

Example Usage:  
Default: [unspecified]

flow\_plan\_selection\_end - - *a Expression field of model*  
MULTIPLE\_FILTER\_AND\_RANK

This expression is called after Flow\_Plan are selected to build the bags. For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer.problem\_period : Bound to the problem period of the currently selected problem.problem\_category : Bound to Problem\_Category of the currently selected problem.active\_strategy : Bound to Active\_Strategy solving problem on this Buffer.

Example Usage:  
Default: [unspecified]

flow\_plan\_selection\_interval - - *a Expression field of model*  
MULTIPLE\_FILTER\_AND\_RANK

This expression returns the Date\_Range over which resolver will look to select Flow\_Plan in order to build bags. For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer.problem\_period : Bound to the problem period of the currently selected problem.problem\_category : Bound to Problem\_Category of the currently selected problem.change\_category : Bound to Strategy\_Change.category for which resolver is building bag.active\_strategy : Bound to Active\_Strategy solving problem on this Buffer.

Example Usage:

Buffer b1 = supply\_chains.find("Sup1").sites.find("Site1").buffers.find(Bufl)  
b1.set\_flow\_plan\_selection\_interval("Date\_Range(problem.start - 1 Week, problem.end)")  
Default: problem\_period

flow\_plan\_filter - - *a Expression field of model*  
MULTIPLE\_FILTER\_AND\_RANK

This expression takes following parameters and if it returns true, the Flow\_Plan will be selected and put into the bag. If it returns false, the Flow\_Plan will not be put into the bag.

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer.problem\_period : Bound to the problem period of the currently selected problem.problem\_category : Bound to Problem\_Category of the currently selected problem.change\_category : Bound to Strategy\_Change.category for which resolver is building bag.active\_strategy : Bound to Active\_Strategy solving problem on this Buffer.flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: true

flow\_plan\_rank - - *a Expression field of model*  
MULTIPLE\_FILTER\_AND\_RANK

This expression returns the 'rank' of the Flow\_Plan which is used by the resolver in selecting the Flow\_Plan. Higher 'rank' means that the Flow\_Plan is more likely to be chosen later from the bag.

For the convenience of OIL expression writer following special variables are available,

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: 1.0

continue\_flow\_plan\_selection - - *a Expression field of model*  
MULTIPLE\_FILTER\_AND\_RANK  
This expression takes following parameters and if it returns true, the selection of Flow\_Plan will continue to be put into the bag. If it returns false, further Flow\_Plans will not be selected.

For the convenience of OIL expression writer following special variables are available.

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: true

flow\_plan\_resize\_quantity\_range - - *a Expression field of model*  
MULTIPLE\_FILTER\_AND\_RANK  
Replan restrictions This expression returns the Quantity\_Range within which the selected Flow\_Plan can be resized by the resolver.

For the convenience of OIL expression writer following special variables are available.

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: [unspecified]

flow\_plan\_move\_restriction - - *a Expression field of model*  
MULTIPLE\_FILTER\_AND\_RANK  
This expression returns the Date\_Range within which the selected Flow\_Plan can be moved by the resolver.

For the convenience of OIL expression writer following special variables are available.

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: [unspecified]

flow\_plan\_split\_restriction - - *a Expression field of model*  
MULTIPLE\_FILTER\_AND\_RANK  
This expression returns the Quantity\_Range within which the selected Flow\_Plan can be split by the resolver for a move

For the convenience of OIL expression writer following special variables are available.

buffer\_plan: : Bound to Buffer\_Plan of this Buffer. problem\_period : Bound to the problem period of the currently selected problem. problem\_category : Bound to Problem\_Category of the currently selected problem change\_category : Bound to Strategy\_Change.category for which resolver is building bag. active\_strategy : Bound to Active\_Strategy solving problem on this Buffer. flow\_plan : Bound to Flow\_Plan while doing bag candidate selection.

Example Usage:  
Default: [unspecified]

**FIXED\_QUANTITY\_FENCED -- a flow\_policy extension of model Buffer**

A **FIXED\_QUANTITY\_FENCED** Buffer issues replenishment Operations all with the same 'quantity'. The Operations are planned to arrive such that the Buffer's on\_hand balance never drops below 'min\_on\_hand'. This policy is often called "Fixed Order Quantity (FOQ)".

This is the same as **FIXED\_QUANTITY** except that it has a 'fence' Horizon\_Date field which computes a Date at and after which a different fixed Quantity 'after\_fence\_quantity' is used, and a different 'after\_fence\_excess\_on\_hand'. This is used for more rough-cut planning a certain amount of time out in the horizon.

Up to 'excess\_on\_hand' more than the minimum required is allowed in the Buffer. By setting 'excess\_on\_hand' larger than zero, planning instability is reduced, and planning speed is increased. However, keeping it smaller minimizes excess inventory.

The **FIXED\_QUANTITY\_FENCED** model has fields that references these models :  
Operation.

**min\_on\_hand -- a Quantity field of model FIXED\_QUANTITY\_FENCED**

The minimum on\_hand balance that should be present in this Buffer. This Quantity is converted to the 'unit' of this Buffer.

Default: 0

**min\_time -- a Time field of model FIXED\_QUANTITY\_FENCED**

Items are planned such that they arrive at least 'min\_time' earlier than otherwise needed. This may be used as a safety time -- the next 'min\_time' of consumption will be maintained in the Buffer. 'min\_time' is represented internally as seconds so if specified as days, weeks, years, its will be converted to seconds. This needs to be considered around daylight savings time boundaries.

Default: 0

**excess\_on\_hand -- a Quantity field of model FIXED\_QUANTITY\_FENCED**

Up to this Quantity more than the minimum required is allowed in this Buffer. Setting this above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels. This Quantity is converted to the 'unit' of this Buffer.

Default: 0

**quantity -- a Quantity field of model FIXED\_QUANTITY\_FENCED**  
The fixed Quantity of the Item to be produced by each 'producing\_operation' planned before and not including 'fence' by this **FIXED\_QUANTITY\_FENCED** Buffer. This Quantity is converted to the 'preferred\_measure' of this Buffer. The default is "1", which means 1 unit of the Buffer.

Default: 1

**fence -- a Horizon\_Date field of model FIXED\_QUANTITY\_FENCED**

The horizon-relative Date at and after which this flow\_policy switches to coarser planning estimates. At and after the fence Date it begins using the 'after\_fence\_quantity' instead of 'quantity' and 'after\_fence\_excess\_on\_hand' instead of 'excess\_on\_hand'.  
Default: oo

**after\_fence\_excess\_on\_hand -- a Quantity field of model FIXED\_QUANTITY\_FENCED**

Up to this Quantity more than the minimum required is allowed in this Buffer. Setting this above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels. This Quantity is converted to the 'unit' of this Buffer.  
Default: 0

**after\_fence\_quantity -- a Quantity field of model FIXED\_QUANTITY\_FENCED**  
The fixed Quantity of the Item to be produced by each 'producing\_operation' planned at and after 'fence' by this **FIXED\_QUANTITY\_FENCED** Buffer. This Quantity is converted to the 'preferred\_measure' of this Buffer. The default is "1", which means 1 unit of the Buffer.

Default: 1

**producing\_operation -- a Operation field of model FIXED\_QUANTITY\_FENCED**

The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field\_Error for this field to be [unspecified], the default, if any Buffer\_Plan needs to issue a producing Operation\_Plan.

Default: [unspecified]

**supplying\_operation -- a Operation field of model FIXED\_QUANTITY\_FENCED**

Obsolete! This field has been renamed 'producing\_operation' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Default: [unspecified]

Extensions	FIXED_TIME Extension
------------	----------------------

Properties:    obsolete=True

**FIXED\_TIME** -- *a flow\_policy extension of model Buffer*

A FIXED\_TIME Buffer is replenished with enough to go at least 'produce\_time' without needing another producing operation. Producing operations are planned such that the Buffer's on\_hand balance never drops below 'min\_on\_hand'. This policy is often called "Periodic Order Quantity (POQ)" or "Fixed Period Ordering".

Up to 'excess\_on\_hand' more than the minimum required is allowed in the Buffer. By setting 'excess\_on\_hand' larger than zero, planning instability is reduced, and planning speed is increased. However, keeping it smaller minimizes excess inventory.

The FIXED\_TIME model has fields that references these models :  
Operation.

**produce\_time** -- *a Time field of model FIXED\_TIME*

The fixed minimum Time desired between 'producing\_operation's by this FIXED\_TIME Buffer. Each 'producing\_operation' is issued for enough to go at least 'produce\_time' without dropping below 'min\_on\_hand'. 'produce\_time' is represented internally as seconds so if specified as days, weeks, years, its will be converted to seconds. This needs to be considered around daylight savings time boundaries.

Default:    0

**min\_on\_hand** -- *a Quantity field of model FIXED\_TIME*

The minimum on\_hand balance that should be present in this Buffer. This Quantity is converted to the 'unit' of this Buffer.

Default:    0

**min\_time** -- *a Time field of model FIXED\_TIME*

Items are planned such that they arrive at least 'min\_time' earlier than otherwise needed. This may be used as a safety time -- the next 'min\_time' of consumption will be maintained in the Buffer. 'min\_time' is represented internally as seconds so if specified as days, weeks, years, its will be converted to seconds. This needs to be considered around daylight savings time boundaries.

Default:    0

**excess\_on\_hand** -- *a Quantity field of model FIXED\_TIME*

Up to this Quantity more than the minimum required is allowed in this Buffer. Setting this above zero reduces instability, and speeds planning. Setting it smaller decreases excess inventory levels. This Quantity is converted to the 'unit' of this Buffer.

Default:    0

Extensions	LFL_SIMPLE Extension
------------	----------------------

**producing\_operation** -- *a Operation field of model FIXED\_TIME*  
The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field\_Error for this field to be [unspecified], the default, if any Buffer\_Plan needs to issue a producing Operation\_Plan.  
Default:    [unspecified]

**supplying\_operation** -- *a Operation field of model FIXED\_TIME*

Obsolete! This field has been renamed 'producing\_operation' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Default:    [unspecified]  
Properties:    obsolete=True

**supply\_time** -- *a Time field of model FIXED\_TIME*

Obsolete! This field has been renamed 'produce\_time' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Default:    0  
Properties:    obsolete=True

**LFL\_SIMPLE** -- *a flow\_policy extension of model Buffer*

A LFL\_SIMPLE Buffer uses a simple Lot-for-Lot order policy: for each Operation that consumes from the Buffer, a 'producing\_operation' with the same Quantity is issued upstream to fill the Buffer.

Excess material is never planned, but if excess material does appear with a planned 'producing\_operation', the downstream Operation will be resized to match.

If an additional Operation\_Plan produces into the Buffer, then a 'consuming\_operation' (if not [unspecified]) will be generated to consume that excess material. If the 'consuming\_operation' is [unspecified], then the excess will remain as on\_hand until a downstream Operation consumes that amount. In that way the 1:1 upstream:downstream is maintained.

The LFL\_SIMPLE model has fields that references these models :  
Operation.

**producing\_operation** - - *a Operation field of model LFL\_SIMPLE*  
The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field\_Error for this field to be [unspecified], the default, if any Buffer\_Plan needs to issue a producing Operation\_Plan.  
Default: [unspecified]

**consuming\_operation** - - *a Operation field of model LFL\_SIMPLE*  
The Operation to be created in order to consume additional flow from this Buffer as needed due to additional producing Operation\_Plans.  
Default: [unspecified]

**supplying\_operation** - - *a Operation field of model LFL\_SIMPLE*  
Obsolete! This field has been renamed 'producing\_operation' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.  
Default: [unspecified]  
Properties: obsolete=True

**LFL\_BOUNDED** - - *a flow\_policy extension of model Buffer*

A LFL\_BOUNDED (Lot For Lot, Bounded) Buffer issues a single 'producing\_operation' upstream, bounded in size by 'quantity\_range', in order to satisfy each downstream Operation that consumes from the Buffer.

If the downstream Operation consumes more than 'quantity\_range.max', then it is resized to 'quantity\_range.max'. If the downstream Operation consumes less than 'quantity\_range.min', the downstream Operation will be resized to 'quantity\_range.min'.

If an upstream Operation\_Plan produces excess material, the downstream Operation\_Plan will be resized to match it. If an additional Operation\_Plan or if excess material otherwise appears in the Buffer, then it will be used to reduce the quantity of one or more 'producing\_operation's. No on\_hand is intentionally maintained.

If an additional Operation\_Plan produces into the Buffer, then a 'consuming\_operation' (if not [unspecified]) will be generated to consume that excess material. If the 'consuming\_operation' is [unspecified], then the excess will remain as on\_hand until a downstream Operation consumes that amount. In that way, the 1:1 upstream:downstream relationship is maintained.

At and after the 'rough\_fence' Horizon\_Date, this flow\_policy uses 'rough\_quantity\_range' rather than 'quantity\_range'. Generally 'rough\_quantity\_range' will be somewhat larger, reducing the computational detail in more distant plans.

If 'rough\_quantity\_range' is from 0 to infinite (the default), then this flow\_policy behaves like LFL\_SIMPLE at and after 'rough\_fence'. If 'quantity\_range' is also from 0 to infinite, then this flow\_policy behaves completely like LFL\_SIMPLE.

If 'quantity\_range' is set to a single value (min == max), then all producing and consuming Operation\_Plans will be resized to that value.

The LFL\_BOUNDED model has fields that references these models :  
Operation.

**quantity\_range** - - *a Quantity\_Range field of model LFL\_BOUNDED*  
The quantity range of items of this Buffer for which to create a 'producing\_operation'.

If the consuming lot is for a Quantity outside this range, it will be resized to fit within this range.

This Quantity is converted to the unit of this Buffer.

If set to infinite, the default, this flow\_policy behaves like LFL\_SIMPLE.  
Default: [0, oo]

**rough\_fence** - - *a Horizon\_Date field of model LFL\_BOUNDED*  
At and after this Horizon\_Date, 'rough\_quantity\_range' is used rather than 'quantity\_range'. Generally 'rough\_quantity\_range' will be a somewhat larger range, reducing the computational detail of more distant plans.  
Default: infinite future

**rough\_quantity\_range** - - *a Quantity\_Range field of model LFL\_BOUNDED*  
The quantity range of items of this Buffer for which to create a 'producing\_operation' at and after the 'rough\_fence'. See 'quantity\_range' for more detail. Generally, this value is a larger range than 'quantity\_range', reducing the computational detail of more distant plans.

If set from 0 to infinite, the default, this flow\_policy behaves like LFL\_SIMPLE at and after the 'rough\_fence'.

Default: [0, 00]

**producing\_operation** - - *a Operation field of model LFL\_BOUNDED*  
The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field\_Error for this field to be [unspecified], the default, if any Buffer\_Plan needs to issue a producing Operation\_Plan.  
Default: [unspecified]

**consuming\_operation** - - *a Operation field of model LFL\_BOUNDED*  
The Operation to be created in order to consume additional flow from this Buffer as needed due to additional producing Operation\_Plans.  
Default: [unspecified]

**supplying\_operation** - - *a Operation field of model LFL\_BOUNDED*  
Obsolete! This field has been renamed 'producing\_operation' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.

Default: [unspecified]  
Properties: obsolete=True

**MLFL\_BOUNDED** - - *a flow\_policy extension of model Buffer*

A MLFL\_BOUNDED (Many-Lots For Lot, Bounded) Buffer issues one or more 'producing\_operation's upstream, bounded in size by 'quantity\_range', in order to satisfy each downstream Operation that consumes from the Buffer.

If the downstream Operation consumes more than 'quantity\_range.max', then multiple 'producing\_operation's will be needed. In such cases, the flow\_policy attempts to keep the number of Operations as small as possible, and their Quantities relatively even.

For example, if 'quantity\_range.max' is 100, and a downstream Operation consumes 240, three 'producing\_operation's each with Quantity 80 is ideal. However, if the Quantity is later adjusted to 250, only one of the upstream orders will be adjusted to 90 (not each to 83, 83, and 84) in order to minimize volatility. Another increase from 250 to 260 will likely increase one of the other orders from 80 to 90 to keep the Quantities reasonably even.

If the downstream Operation consumes less than 'quantity\_range.min', the downstream Operation will be resized to 'quantity\_range.min'. Similarly, if excess material is produced by a 'producing\_operation', then the downstream Operation will be resized.

If an additional Operation\_Plan produces into the Buffer, then a 'consuming\_operation' (if not [unspecified]) will be generated to consume that excess material. If the 'consuming\_operation' is [unspecified], then one of the downstream operations (which is not defined) will be resized to consume the excess. In that way the N:1 upstream:downstream relationship is maintained, and no on\_hand balance is planned.

At and after the 'rough\_fence' Horizon\_Date, this flow\_policy uses 'rough\_quantity\_range' rather than 'quantity\_range'. Generally 'rough\_quantity\_range' will be somewhat larger, reducing the computational detail in more distant plans.

If 'rough\_quantity\_range' is from 0 to infinite (the default), then this flow\_policy behaves like LFL\_SIMPLE at and after 'rough\_fence'. If 'quantity\_range' is also from 0 to infinite, then this flow\_policy behaves completely like LFL\_SIMPLE.

The MLFL\_BOUNDED model has fields that references these models :  
Operation.

**quantity\_range** - - *a Quantity\_Range field of model MLFL\_BOUNDED*  
The quantity range of items of this Buffer for which to create a 'producing\_operation'.

If the consuming lot is for a Quantity greater than this then the lots produced by the 'producing\_operation' will be broken into the largest balanced orders that are less than or equal to this Quantity. For example, if this is 100, and an order for 240 comes in, 3 orders will be issued, each with Quantity 80.

If the consuming lot is for a Quantity less than this then it will be resized

This Quantity is converted to the unit of this Buffer.

If set to infinite, the default, this flow\_policy behaves like LFL\_SIMPLE.  
Default: [0, 00]

**rough\_fence** - - *a Horizon\_Date field of model MLFL\_BOUNDED*  
At and after this Horizon\_Date, 'rough\_quantity\_range' is used rather than 'quantity\_range'. Generally 'rough\_quantity\_range' will be a somewhat larger range, reducing the computational detail of more distant plans.  
Default: infinite future

Extensions

stocking\_policy extensions of model Buffer

**rough\_quantity\_range** - - *a Quantity\_Range field of model MLFL\_BOUNDED*  
The quantity range of items of this Buffer for which to create a 'producing\_operation' at and after the 'rough\_fence'. See 'quantity\_range' for more detail. Generally, this value is a larger range than 'quantity\_range', reducing the computational detail of more distant plans.

If set from 0 to infinite, the default, this flow\_policy behaves like LFL\_SIMPLE at and after the 'rough\_fence'.  
Default: [0, oo]

**producing\_operation** - - *a Operation field of model MLFL\_BOUNDED*  
The Operation to be created in order to produce additional flow into this Buffer as needed due to consuming Operations.

It is a Field\_Error for this field to be [unspecified], the default, if any Buffer\_Plan needs to issue a producing Operation\_Plan.  
Default: [unspecified]

**consuming\_operation** - - *a Operation field of model MLFL\_BOUNDED*  
The Operation to be created in order to consume additional flow from this Buffer as needed due to additional producing Operation\_Plans.  
Default: [unspecified]

**supplying\_operation** - - *a Operation field of model MLFL\_BOUNDED*  
Obsolete! This field has been renamed 'producing\_operation' in an effort to have more consistent and less ambiguous naming. This field will be eliminated in a future release.  
Default: [unspecified]  
Properties: obsolete=True

10.16.2 stocking\_policy extensions of model Buffer

MANUAL -- a stocking\_policy extension of model Buffer

A Manual stocking\_policy implements no relevant stocking constraints. The user specifies the min\_on\_hand and min\_time (in numbers but not a formula) in the flow\_policy.

CALENDAR -- a stocking\_policy extension of model Buffer

Extensions

CALENDAR Extension

A CALENDAR stocking\_policy computes safety stock (min\_on\_hand and min\_time) based on mean and standard deviation of lead time and demand. Other inputs are the replenishment interval or quantity. It computes safety stock (min\_on\_hand and min\_time) one buffer at a time i.e. using local information. It is assumed that the demand on this buffer is normally distributed.

Currently CALENDAR stocking\_policy is only implemented for the PRODUCING\_FLOW\_CALENDAR flow\_policy. Note that the user can either bias or override the automatically computed safety stock by specifying a custom formula. See the user\_bias field description for details.

Most of the fields in the 'CALENDAR' stocking\_policy can be specified in a Calendar, so they can change over time.

The CALENDAR model has fields that references these models :  
Calendar.

**units\_of\_safety\_stock** - - *a Safety\_Stock\_Units field of model CALENDAR*  
Specifies if min\_on\_hand and/or min\_time should be automatically computed.

If QUANTITY then the safety stock will be calculated in quantity of units and only min\_on\_hand will be automatically set by the stocking\_policy.

If TIME then the safety stock will be calculated in days of coverage and only min\_time will be automatically set by the stocking\_policy.

If QUANTITY\_TIME then the safety stock will be calculated in both quantity of units and days of coverage. First min\_on\_hand will be computed and then converted to min\_time.  
Default: QUANTITY

**user\_bias** - - *a Expression field of model CALENDAR*

Used for either biasing or overriding the automatically computed safety stock by specifying a custom formula in terms of an OIL expression. If min\_on\_hand and min\_time are both automatically calculated then the bias is first applied to min\_on\_hand and then the biased min\_on\_hand is converted into min\_time.

When the expression executes:  
buffer' is set to the buffer whose stocking policy is being calculated.  
min\_on\_hand' is set to the min\_on\_hand calculated  
'start\_date' is the starting date where the min\_on\_hand applies  
'end\_date' is the ending date where the min\_on\_hand applies

Extensions	CALENDAR Extension
------------	--------------------

'mean\_lead\_time' A Time  
'std\_dev\_lead\_time' A Time  
'mean\_demand' A Quantity  
'std\_dev\_demand' A Quantity

The result from the expression will be used as the min\_on\_hand for the buffer's flow\_policy from start\_date to end\_date.

Examples:

1. Additive Bias:

Setting user\_bias to min\_on\_hand + 10' will first compute min\_on\_hand using the pre-defined formula and then add 10 to that number.

2. Multiplicative Bias:

Setting user\_bias to min\_on\_hand \* 1.2' will first compute min\_on\_hand using the pre-defined formula and then increase it 1.2 fold.

3. Time-phased Bias:

Setting user\_bias to if (and (start\_date >= "1/1/97", end\_date < "4/1/97"), min\_on\_hand \* 1.2, min\_on\_hand \* 1.1)' will first compute min\_on\_hand using the pre-defined formula and then increase it 1.2 fold for the first quarter of 1997 while increase it 1.1 fold for other times.

4. Overriding Pre-defined Formula:

Setting user\_bias to 'mean\_demand \* 0.25' sets the min\_on\_hand to the 1/4th of the mean\_demand. Note that the safety stock computation due to the pre-defined formula is by-passed here with a user-specified formula.

Default: min\_on\_hand

customer\_service\_level - - a Percentage field of model CALENDAR

The expected level of service from this buffer. This field can not be set to 100 as that would result in infinite safety stock.

Default: 99

default\_mean\_demand - - a Quantity field of model CALENDAR

Mean demand is the average demand rate (in units/days) put on this buffer by buffers immediately downstream to it.

Extensions	CALENDAR Extension
------------	--------------------

Default: 0

default\_mean\_demand\_time\_bucket - - a Time field of model CALENDAR

Specifies the time bucket over which mean\_demand is defined.

Default: 1 Day

mean\_demand\_calendar - - a Calendar field of model CALENDAR

Time-phased average demand rate (in units/days).

Default: [unspecified]

default\_standard\_deviation\_demand - - a Quantity field of model CALENDAR

Standard deviation of demand is the variation in demand put on this buffer by buffers immediately downstream to it.

Default: 0

standard\_deviation\_demand\_calendar - - a Calendar field of model CALENDAR

Time-phased standard deviation of demand.

Default: [unspecified]

default\_mean\_lead\_time - - a Time field of model CALENDAR

Mean lead time is the average time to produce/supply material into this buffer from buffers immediately upstream to it.

Default: 0

mean\_lead\_time\_calendar - - a Calendar field of model CALENDAR

Time-phased mean lead time.

Default: [unspecified]

default\_standard\_deviation\_lead\_time - - a Time field of model CALENDAR

Standard deviation of lead time is the variation in producing/supplying material into this buffer from buffers immediately upstream to it.

Default: 0

standard\_deviation\_lead\_time\_calendar - - a Calendar field of model CALENDAR

DAR

Time-phased standard deviation of lead time.

Default: [unspecified]

safety\_factor, safety\_factor(Date) - - a Quantity field of model CALENDAR

Safety factor scales the supply and demand variations in accordance to the customer service level expectations.



<i>Extensions</i>	<i>CALENDAR Extension</i>
-------------------	---------------------------

Properties:    Export-Only Field

demand\_safety\_stock, demand\_safety\_stock (Date)    - - *a* Quantity field of model  
CALENDAR  
Safety stock due to demand variation only.  
Properties:    Export-Only Field

supply\_safety\_stock, supply\_safety\_stock (Date)    - - *a* Quantity field of model  
CALENDAR  
Safety stock due to supply variation only.  
Properties:    Export-Only Field

<i>Extensions</i>	<i>Buffer_Problem_Detector Extensions</i>
-------------------	---

10.17    Buffer\_Problem\_Detector Extensions

10.17.1    detector extensions of model Buffer\_Problem\_Detector

### 10.18 Buffer\_Plan Extensions

#### 10.18.1 Flow\_policy extensions of model Buffer\_Plan

##### BUCKETED\_NESTED\_SORT -- a flow\_policy extension of model Buffer\_Plan

A BUCKETED\_NESTED\_SORT Buffer manages the Flow\_Plans in buckets of time, as specified by the horizon. See the corresponding 'flow\_policy' of the Buffer model for more information.

The BUCKETED\_NESTED\_SORT model has these submodels:

Sorted\_Bucket.

The BUCKETED\_NESTED\_SORT model has fields that references these models:  
Sorted\_Bucket.

sorted\_buckets -- a list of Sorted\_Bucket submodels of model  
BUCKETED\_NESTED\_SORT

allocate\_excess -- a Void field of model BUCKETED\_NESTED\_SORT

This command allocates unused material in the Buffer\_Plan by creating forecast Requests for the 'default\_product\_root' of the Buffer.

Properties: command=True Export-Only Field

##### PRODUCING\_FLOW\_CALENDAR -- a flow\_policy extension of model Buffer\_Plan

A PRODUCING\_FLOW\_CALENDAR Buffer maintains a 'min\_time' safety time and a 'min\_on\_hand' safety quantity. See the corresponding 'flow\_policy' of the Buffer model for more details.

max\_target\_profile, max\_target\_profile (Date\_Range) -- a

List(Profile\_Quantity) field of model PRODUCING\_FLOW\_CALENDAR

A profile or a list of changes to max allowable Quantity's during the specified finite Date\_Range. Up to 'max\_target' Quantity is allowed in this buffer. If the 'Buffer\_Plan.on\_hand' Quantity goes above 'Buffer\_Plan.max\_target' Quantity, RHYTHM will detect EXCESS\_ON\_HAND problem.

Properties: command=True Export-Only Field

max\_target (Date), max\_target (Date\_Range, Logical) -- a Quantity field of model PRODUCING\_FLOW\_CALENDAR

Maximum target Quantity allowed in this BASIC buffer. If the 'Buffer\_Plan.on\_hand' Quantity goes above 'Buffer\_Plan.max\_target' Quantity, RHYTHM will detect EXCESS\_ON\_HAND problem.

If a Date is passed, this returns the 'max\_target' Quantity allowed as of that Date in this buffer.

If a Date\_Range is passed, this returns the smallest 'max\_target' Quantity allowed during that Date\_Range in this buffer.

If a Date\_Range and a Logical is passed, this returns the smallest 'max\_target' Quantity allowed during that Date\_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest 'max\_target' Quantity allowed during that Date\_Range.

The Quantity is converted to the 'unit' of this buffer.

Properties: Export-Only Field

safety\_target\_profile, safety\_target\_profile (Date\_Range) -- a

List(Profile\_Quantity) field of model PRODUCING\_FLOW\_CALENDAR

A profile or a list of changes to the 'safety\_target' Quantity during the specified Date\_Range. This 'safety\_target' Quantity is the max of 'Buffer.min\_on\_hand' and the Quantity required to cover the next 'min\_time' of consumption in this Buffer.

For example, consider a Buffer with 'min\_on\_hand' is set to "100", 'min\_time' is set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety\_target\_profile' will have three entries, first entry with quantity 100 until 4/1, second entry with quantity 200 starting on 4/1 and third entry with quantity 100 starting on 4/15. In the second period "200" will be maintained in order to cover the next "2 weeks" of out flows.

Properties: command=True Export-Only Field

safety\_target (Date), safety\_target (Date\_Range, Logical) -- a Quantity field of model PRODUCING\_FLOW\_CALENDAR

Safety target is a minimum Quantity this BASIC Buffer is required to maintain based on existing plan. Mathematically, 'safety\_target' Quantity is the max of 'Buffer.min\_on\_hand' and the Quantity required to cover the next 'Buffer.min\_time' of consumption in this Buffer.

For example, consider a Buffer with min\_on\_hand set to "100" and min\_lime set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the safety\_target Quantity on date 4/10 will be 200.

If a Date is passed, this returns the safety\_target Quantity required to be maintained as of that Date considering the existing plan.

If a Date\_Range is passed, this returns the smallest safety\_target Quantity required to be maintained during that Date\_Range considering the existing plan.

If a Date\_Range and a Logical is passed, this returns the smallest safety\_target Quantity required to be maintained during that Date\_Range considering the existing plan if the Logical is True. If the Logical is False, this returns the largest safety\_target Quantity required to be maintained during that Date\_Range.

The Quantity is converted to the unit of this buffer.

Properties:    Export-Only Field

cycle\_on\_hand\_profile, cycle\_on\_hand (Date\_Range) -- a

List(Profile\_Quantity) field of model PRODUCING\_FLOW\_CALENDAR

A profile or a list of changes to cycle\_on\_hand (max(0,0, Buffer\_Plan.on\_hand - Buffer\_Plan.safety\_target)) Quantity's planned during the specified finite Date\_Range in this buffer.

Properties:    command=True    Export-Only Field

cycle\_on\_hand (Date), cycle\_on\_hand (Date\_Range, Logical) -- a Quantity

field of model PRODUCING\_FLOW\_CALENDAR

Cycle\_on\_hand (max(0,0, Buffer\_Plan.on\_hand - Buffer\_Plan.safety\_target)) Quantity planned in this buffer.

If a Date is passed, this returns the cycle\_on\_hand Quantity planned as of that Date in this buffer.

If a Date\_Range is passed, this returns the smallest cycle\_on\_hand Quantity planned during that Date\_Range in this buffer.

If a Date\_Range and a Logical is passed, this returns the smallest cycle\_on\_hand Quantity planned during that Date\_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest max\_target Quantity planned during that Date\_Range.

The Quantity is converted to the unit of this buffer.

Properties:    Export-Only Field

excess\_on\_hand\_profile, excess\_on\_hand\_profile (Date\_Range) -- a

List(Profile\_Quantity) field of model PRODUCING\_FLOW\_CALENDAR

A profile or a list of changes to excess\_on\_hand Quantity during the specified finite Date\_Range. This is the Buffer\_Plan.on\_hand Quantity above allowable Buffer\_Plan.max\_target Quantity. Mathematically, 'excess\_on\_hand' is computed using formula, max(0,0, Buffer\_Plan.on\_hand - Buffer\_Plan.max\_target).

Properties:    command=True    Export-Only Field

excess\_on\_hand (Date), excess\_on\_hand (Date\_Range, Logical) -- a Quantity

field of model PRODUCING\_FLOW\_CALENDAR

Excess\_on\_hand Quantity planned above the allowable Buffer.excess\_on\_hand (will be renamed to Buffer.max\_cycle\_quantity) Quantity in this buffer. Mathematically, 'excess\_on\_hand' is computed using formula, max(0,0, Buffer\_Plan.on\_hand - Buffer\_Plan.max\_target).

If a Date is passed, this returns the excess\_on\_hand Quantity planned as of that Date.

If a Date\_Range is passed, this returns the smallest excess\_on\_hand Quantity planned during that Date\_Range.

If a Date\_Range and a Logical is passed, this returns the smallest excess\_on\_hand Quantity planned during that Date\_Range, if the Logical is True. If the Logical is False, this returns the largest excess\_on\_hand Quantity planned during that Date\_Range.

The Quantity is converted to the unit of this buffer.

Properties:    Export-Only Field

low\_on\_hand\_profile, low\_on\_hand\_profile (Date\_Range) -- a

List(Profile\_Quantity) field of model PRODUCING\_FLOW\_CALENDAR

A profile or a list of changes to the low\_on\_hand Quantity during the specified Date\_Range. Mathematically, low\_on\_hand is computed using following formula, (Buffer\_Plan.safety\_target(d) - Buffer\_Plan.on\_hand(d)).

Properties:    command=True    Export-Only Field

`low_on_hand (Date), low_on_hand (Date_Range, Logical) -- a Quantity field of model PRODUCING_FLOW_CALENDAR`  
It is the 'on\_hand' Quantity planned below the minimum 'safety\_target' Quantity. Mathematically, 'low\_on\_hand' is computed using following formula,  
(Buffer\_Plan.safety\_target - Buffer\_Plan.on\_hand).

If a Date is passed, this returns the 'low\_on\_hand' Quantity planned as of that Date.

If a Date\_Range is passed, this returns the smallest 'low\_on\_hand' Quantity planned during that Date\_Range.

If a Date\_Range and a Logical is passed, this returns the smallest 'low\_on\_hand' Quantity planned during that Date\_Range, if the Logical is True. If the Logical is False, this returns the largest 'low\_on\_hand' Quantity planned during that Date\_Range.

The Quantity is converted to the 'unit' of this buffer.  
Properties:    Export-Only Field

**PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK -- a flow\_policy extension of model Buffer\_Plan**

Just like Producing Flow Calendar Flow Policy

`max_target_profile, max_target_profile (Date_Range) -- a List(Profile_Quantity) field of model PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK`  
A profile or a list of changes to max allowable Quantity's during the specified finite Date\_Range. Up to 'max\_target' Quantity is allowed in this buffer. If the Buffer\_Plan.on\_hand' Quantity goes above Buffer\_Plan.max\_target' Quantity, RHYTHM will detect EXCESS\_ON\_HAND problem.  
Properties:    command=True    Export-Only Field

`max_target (Date), max_target (Date_Range, Logical) -- a Quantity field of model PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK`  
Maximum target Quantity allowed in this BASIC buffer. If the Buffer\_Plan.on\_hand' Quantity goes above Buffer\_Plan.max\_target' Quantity, RHYTHM will detect EXCESS\_ON\_HAND problem.

If a Date is passed, this returns the 'max\_target' Quantity allowed as of that Date in this buffer.

If a Date\_Range is passed, this returns the smallest 'max\_target' Quantity allowed during that Date\_Range in this buffer.

If a Date\_Range and a Logical is passed, this returns the smallest 'max\_target' Quantity allowed during that Date\_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest 'max\_target' Quantity allowed during that Date\_Range.

The Quantity is converted to the 'unit' of this buffer.  
Properties:    Export-Only Field

`safety_target_profile, safety_target_profile (Date_Range) -- a List(Profile_Quantity) field of model PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK`  
A profile or a list of changes to the 'safety\_target' Quantity during the specified Date\_Range. This 'safety\_target' Quantity is the max of Buffer.min\_on\_hand and the Quantity required to cover the next 'min\_time' of consumption in this Buffer.

For example, consider a Buffer with 'min\_on\_hand' is set to "100", 'min\_time' is set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety\_target\_profile' will have three entries, first entry with quantity 100 until 4/1, second entry with quantity 200 starting on 4/1 and third entry with quantity 100 starting on 4/15. In the second period "200" will be maintained in order to cover the next "2 weeks" of out flows.

Properties:    command=True    Export-Only Field

`safety_target (Date), safety_target (Date_Range, Logical) -- a Quantity field of model PRODUCING_FLOW_CALENDAR_FILTER_AND_RANK`  
Safety target is a minimum Quantity this BASIC Buffer is required to maintain based on existing plan. Mathematically, 'safety\_target' Quantity is the max of Buffer.min\_on\_hand and the Quantity required to cover the next 'Buffer.min\_time' of consumption in this Buffer.

For example, consider a Buffer with 'min\_on\_hand' set to "100" and 'min\_time' set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety\_target' Quantity on date 4/10 will be 200.

If a Date is passed, this returns the 'safety\_target' Quantity required to be maintained as of that Date considering the existing plan.

If a Date\_Range is passed, this returns the smallest 'safety\_target' Quantity required to be maintained during that Date\_Range considering the existing plan.

If a *Date\_Range* and a *Logical* is passed, this returns the smallest 'safety\_target' Quantity required to be maintained during that *Date\_Range* considering the existing plan if the *Logical* is *True*. If the *Logical* is *False*, this returns the largest 'safety\_target' Quantity required to be maintained during that *Date\_Range*.

The Quantity is converted to the 'unit' of this buffer:

Properties:    Export-Only Field

*cycle\_on\_hand\_profile*, *cycle\_on\_hand\_profile* (*Date\_Range*)    -- *a*

List(*Profile\_Quantity*) *field of model*

**PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

A profile or a list of changes to *cycle\_on\_hand* (max(0.0, *Buffer\_Plan.on\_hand - Buffer\_Plan.safety\_target*)) Quantity's planned during the specified finite *Date\_Range* in this buffer.

Properties:    command=*True*    Export-Only Field

*cycle\_on\_hand* (*Date*), *cycle\_on\_hand* (*Date\_Range*, *Logical*)    -- *a* Quantity *field of model* **PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**  
*Cycle\_on\_hand* (max(0.0, *Buffer\_Plan.on\_hand - Buffer\_Plan.safety\_target*)) Quantity planned in this buffer.

If a *Date* is passed, this returns the 'cycle\_on\_hand' Quantity planned as of that *Date* in this buffer.

If a *Date\_Range* is passed, this returns the smallest 'cycle\_on\_hand' Quantity planned during that *Date\_Range* in this buffer.

If a *Date\_Range* and a *Logical* is passed, this returns the smallest 'cycle\_on\_hand' Quantity planned during that *Date\_Range* in this buffer if the *Logical* is *True*. If the *Logical* is *False*, this returns the largest 'max\_target' Quantity planned during that *Date\_Range*.

The Quantity is converted to the 'unit' of this buffer:

Properties:    Export-Only Field

*excess\_on\_hand\_profile*, *excess\_on\_hand\_profile* (*Date\_Range*)    -- *a*

List(*Profile\_Quantity*) *field of model*

**PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

A profile or a list of changes to *excess\_on\_hand* Quantity during the specified finite *Date\_Range*. This is the *Buffer\_Plan.on\_hand* Quantity above allowable *Buffer\_Plan.max\_target* Quantity. Mathematically, 'excess\_on\_hand' is computed using formula, max(0.0, *Buffer\_Plan.on\_hand - Buffer\_Plan.max\_target*).

Properties:    command=*True*    Export-Only Field

*excess\_on\_hand* (*Date*), *excess\_on\_hand* (*Date\_Range*, *Logical*)    -- *a* Quantity *field of model* **PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

*Excess\_on\_hand* Quantity planned above the allowable *Buffer.excess\_on\_hand* (will be renamed to *Buffer.max\_cycle\_quantity*) Quantity in this buffer. Mathematically, 'excess\_on\_hand' is computed using formula, max(0.0, *Buffer\_Plan.on\_hand - Buffer\_Plan.max\_target*).

If a *Date* is passed, this returns the 'excess\_on\_hand' Quantity planned as of that *Date*.

If a *Date\_Range* is passed, this returns the smallest 'excess\_on\_hand' Quantity planned during that *Date\_Range*.

If a *Date\_Range* and a *Logical* is passed, this returns the smallest 'excess\_on\_hand' Quantity planned during that *Date\_Range*. If the *Logical* is *True*. If the *Logical* is *False*, this returns the largest 'excess\_on\_hand' Quantity planned during that *Date\_Range*.

The Quantity is converted to the 'unit' of this buffer:

Properties:    Export-Only Field

*low\_on\_hand\_profile*, *low\_on\_hand\_profile* (*Date\_Range*)    -- *a*

List(*Profile\_Quantity*) *field of model*

**PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

A profile or a list of changes to the 'low\_on\_hand' Quantity during the specified *Date\_Range*. Mathematically, 'low\_on\_hand' is computed using following formula, (*Buffer\_Plan.safety\_target*(d) - *Buffer\_Plan.on\_hand*(d)).

Properties:    command=*True*    Export-Only Field

*low\_on\_hand* (*Date*), *low\_on\_hand* (*Date\_Range*, *Logical*)    -- *a* Quantity *field of model* **PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK**

It is the 'on\_hand' Quantity planned below the minimum 'safety\_target' Quantity. Mathematically, 'low\_on\_hand' is computed using following formula, (*Buffer\_Plan.safety\_target - Buffer\_Plan.on\_hand*).

If a Date is passed, this returns the 'low\_on\_hand' Quantity planned as of that Date.

If a Date\_Range is passed, this returns the smallest 'low\_on\_hand' Quantity planned during that Date\_Range.

If a Date\_Range and a Logical is passed, this returns the smallest 'low\_on\_hand' Quantity planned during that Date\_Range, if the Logical is True. If the Logical is False, this returns the largest 'low\_on\_hand' Quantity planned during that Date\_Range.

The Quantity is converted to the 'unit' of this buffer.

Properties: Export-Only Field

ON\_HAND\_CALENDAR -- a flow\_policy extension of model Buffer\_Plan

An ON\_HAND\_CALENDAR Buffer maintains a 'min\_time' safety time and a 'min\_on\_hand' safety quantity. See the corresponding 'flow\_policy' of the Buffer model for more details.

capacity (Date\_Range) -- a Quantity field of model ON\_HAND\_CALENDAR  
This returns the gross capacity in the specified date range given the underlying on\_hand\_calendar. Note that in cases where the date\_range does not completely enclose the bucket\_spec of the calendar, the entire capacity of the bucket is available for consumption and is returned.  
Properties: Export-Only Field

ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK -- a flow\_policy extension of model Buffer\_Plan

An ON\_HAND\_CALENDAR Buffer maintains a 'min\_time' safety time and a 'min\_on\_hand' safety quantity. See the corresponding 'flow\_policy' of the Buffer model for more details.

capacity (Date\_Range) -- a Quantity field of model ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK  
This returns the gross capacity in the specified date range given the underlying on\_hand\_calendar. Note that in cases where the date\_range does not completely enclose the bucket\_spec of the calendar, the entire capacity of the bucket is available for consumption and is returned.  
Properties: Export-Only Field

FLOW\_LIMIT\_CALENDAR -- a flow\_policy extension of model Buffer\_Plan

A FLOW\_LIMIT\_CALENDAR Buffer has a behaviour identical to a ON\_HAND\_CALENDAR except that the NEGATIVE\_ON\_HAND problems are in this case called OVER\_FLOW\_LIMIT problems. See the corresponding 'flow\_policy' of the Buffer model for more details.

capacity (Date\_Range) -- a Quantity field of model FLOW\_LIMIT\_CALENDAR  
This returns the gross capacity in the specified date range given the underlying flow\_limit\_calendar. Note that in cases where the date\_range does not completely enclose the bucket\_spec of the calendar, the entire capacity of the bucket is available for consumption and is returned.  
Properties: Export-Only Field

FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK -- a flow\_policy extension of model Buffer\_Plan

A FLOW\_LIMIT\_CALENDAR Buffer has a behaviour identical to a ON\_HAND\_CALENDAR except that the NEGATIVE\_ON\_HAND problems are in this case called OVER\_FLOW\_LIMIT problems. See the corresponding 'flow\_policy' of the Buffer model for more details.

capacity (Date\_Range) -- a Quantity field of model FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK  
This returns the gross capacity in the specified date range given the underlying flow\_limit\_calendar. Note that in cases where the date\_range does not completely enclose the bucket\_spec of the calendar, the entire capacity of the bucket is available for consumption and is returned.  
Properties: Export-Only Field

BASIC -- a flow\_policy extension of model Buffer\_Plan

A BASIC Buffer maintains a 'min\_time' safety time and a 'min\_on\_hand' safety quantity. See the corresponding 'flow\_policy' of the Buffer model for more details.

max\_target\_profile, max\_target\_profile (Date\_Range) -- a List(Profile\_Quantity) field of model BASIC  
A profile or a list of changes to max allowable Quantity's during the specified finite Date\_Range. Up to 'max\_target' Quantity is allowed in this buffer. If the Buffer\_Plan.on\_hand' Quantity goes above 'Buffer\_Plan.max\_target' Quantity, RHYTHM will detect EXCESS\_ON\_HAND problem.  
Properties: command=True Export-Only Field

Extensions	BASIC Extension
------------	-----------------

max\_target (Date), max\_target (Date\_Range, Logical) -- a Quantity field of model BASIC  
Maximum target Quantity allowed in this BASIC buffer. If the Buffer\_Plan.on\_hand' Quantity goes above Buffer\_Plan.max\_target Quantity, RHYTHM will detect EXCESS\_ON\_HAND problem.

If a Date is passed, this returns the 'max\_target' Quantity allowed as of that Date in this buffer.

If a Date\_Range is passed, this returns the smallest 'max\_target' Quantity allowed during that Date\_Range in this buffer.

If a Date\_Range and a Logical is passed, this returns the smallest 'max\_target' Quantity allowed during that Date\_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest 'max\_target' Quantity allowed during that Date\_Range.

The Quantity is converted to the unit of this buffer.  
Properties: Export-Only Field

safety\_target\_profile, safety\_target\_profile (Date\_Range) -- a List(Profile\_Quantity) field of model BASIC  
A profile or a list of changes to the 'safety\_target' Quantity during the specified Date\_Range. This 'safety\_target' Quantity is the max of Buffer.min\_on\_hand and the Quantity required to cover the next 'min\_time' of consumption in this Buffer.

For example, consider a Buffer with 'min\_on\_hand' is set to "100", 'min\_time' is set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety\_target\_profile' will have three entries, first entry with quantity 100 until 4/1, second entry with quantity 200 starting on 4/1 and third entry with quantity 100 starting on 4/15. In the second period "200" will be maintained in order to cover the next "2 weeks" of out flows.  
Properties: command=True Export-Only Field

safety\_target (Date), safety\_target (Date\_Range, Logical) -- a Quantity field of model BASIC  
Safety target is a minimum Quantity this BASIC Buffer is required to maintain based on existing plan. Mathematically, 'safety\_target' Quantity is the max of Buffer.min\_on\_hand and the Quantity required to cover the next 'Buffer.min\_time' of consumption in this Buffer.

Extensions	BASIC Extension
------------	-----------------

For example, consider a Buffer with 'min\_on\_hand' set to "100" and 'min\_time' set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety\_target' Quantity on date 4/10 will be 200.

If a Date is passed, this returns the 'safety\_target' Quantity required to be maintained as of that Date considering the existing plan.

If a Date\_Range is passed, this returns the smallest 'safety\_target' Quantity required to be maintained during that Date\_Range considering the existing plan.

If a Date\_Range and a Logical is passed, this returns the smallest 'safety\_target' Quantity required to be maintained during that Date\_Range considering the existing plan if the Logical is True. If the Logical is False, this returns the largest 'safety\_target' Quantity required to be maintained during that Date\_Range.

The Quantity is converted to the unit of this buffer.  
Properties: Export-Only Field

cycle\_on\_hand\_profile, cycle\_on\_hand\_profile (Date\_Range) -- a List(Profile\_Quantity) field of model BASIC  
A profile or a list of changes to cycle 'on\_hand' (max(0.0, Buffer\_Plan.on\_hand - Buffer\_Plan.safety\_target)) Quantity's planned during the specified finite Date\_Range in this buffer.  
Properties: command=True Export-Only Field

cycle\_on\_hand (Date), cycle\_on\_hand (Date\_Range, Logical) -- a Quantity field of model BASIC  
Cycle 'on\_hand' (max(0.0, Buffer\_Plan.on\_hand - Buffer\_Plan.safety\_target)) Quantity planned in this buffer.

If a Date is passed, this returns the 'cycle\_on\_hand' Quantity planned as of that Date in this buffer.

If a Date\_Range is passed, this returns the smallest 'cycle\_on\_hand' Quantity planned during that Date\_Range in this buffer.

If a Date\_Range and a Logical is passed, this returns the smallest 'cycle\_on\_hand' Quantity planned during that Date\_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest 'max\_target' Quantity planned during that Date\_Range.

**excess\_on\_hand (Date), excess\_on\_hand (Date\_Range, Logical) -- a Quantity field of model BASIC\_FILTER\_AND\_RANK**  
 Excess 'on\_hand' Quantity planned above the allowable 'Buffer.excess\_on\_hand' will be renamed to 'Buffer.max\_cycle\_quantity' Quantity in this buffer. Mathematically, 'excess\_on\_hand' is computed using formula,  $\max(0.0, \text{Buffer\_Plan.on\_hand} - \text{Buffer\_Plan.max\_target})$ .

If a Date is passed, this returns the 'excess\_on\_hand' Quantity planned as of that Date.

If a Date\_Range is passed, this returns the smallest 'excess\_on\_hand' Quantity planned during that Date\_Range.

If a Date\_Range and a Logical is passed, this returns the smallest 'excess\_on\_hand' Quantity planned during that Date\_Range, if the Logical is True. If the Logical is False, this returns the largest 'excess\_on\_hand' Quantity planned during that Date\_Range.

The Quantity is converted to the 'unit' of this buffer.

Properties: Export-Only Field

**low\_on\_hand\_profile, low\_on\_hand\_profile (Date\_Range) -- a**

**List(Profile\_Quantity) field of model BASIC\_FILTER\_AND\_RANK**

A profile or a list of changes to the 'low\_on\_hand' Quantity during the specified Date\_Range. Mathematically, 'low\_on\_hand' is computed using following formula,  $(\text{Buffer\_Plan.safety\_target}(d) - \text{Buffer\_Plan.on\_hand}(d))$ .

Properties: command=True Export-Only Field

**low\_on\_hand (Date), low\_on\_hand (Date\_Range, Logical) -- a Quantity field of model BASIC\_FILTER\_AND\_RANK**

It is the 'on\_hand' Quantity planned below the minimum 'safety\_target' Quantity. Mathematically, 'low\_on\_hand' is computed using following formula,  $(\text{Buffer\_Plan.safety\_target} - \text{Buffer\_Plan.on\_hand})$ .

If a Date is passed, this returns the 'low\_on\_hand' Quantity planned as of that Date.

If a Date\_Range is passed, this returns the smallest 'low\_on\_hand' Quantity planned during that Date\_Range.

If a Date\_Range and a Logical is passed, this returns the smallest 'low\_on\_hand' Quantity planned during that Date\_Range, if the Logical is True. If the Logical is False, this returns the largest 'low\_on\_hand' Quantity planned during that Date\_Range.

The Quantity is converted to the 'unit' of this buffer.

Properties: Export-Only Field

**FIXED\_QUANTITY -- a flow\_policy extension of model Buffer\_Plan**

A **FIXED\_QUANTITY** Buffer maintains a 'min\_time' safety time and a 'min\_on\_hand' safety quantity. See the corresponding 'flow\_policy' of the Buffer model for more details.

**max\_target\_profile, max\_target\_profile (Date\_Range) -- a**

**List(Profile\_Quantity) field of model FIXED\_QUANTITY**

A profile or a list of changes to max allowable Quantity's during the specified finite Date\_Range. Up to 'max\_target' Quantity is allowed in this buffer. If the 'Buffer\_Plan.on\_hand' Quantity goes above 'Buffer\_Plan.max\_target' Quantity, RHYTHM will detect EXCESS\_ON\_HAND problem.

Properties: command=True Export-Only Field

**max\_target (Date), max\_target (Date\_Range, Logical) -- a Quantity field of model FIXED\_QUANTITY**

Maximum target Quantity allowed in this BASIC buffer. If the 'Buffer\_Plan.on\_hand' Quantity goes above 'Buffer\_Plan.max\_target' Quantity, RHYTHM will detect EXCESS\_ON\_HAND problem.

If a Date is passed, this returns the 'max\_target' Quantity allowed as of that Date in this buffer.

If a Date\_Range is passed, this returns the smallest 'max\_target' Quantity allowed during that Date\_Range in this buffer.

If a Date\_Range and a Logical is passed, this returns the smallest 'max\_target' Quantity allowed during that Date\_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest 'max\_target' Quantity allowed during that Date\_Range.

The Quantity is converted to the 'unit' of this buffer.

Properties: Export-Only Field



**safety\_target\_profile, safety\_target\_profile (Date\_Range) -- a**  
**List(Profile\_Quantity) field of model FIXED\_QUANTITY**

A profile or a list of changes to the 'safety\_target' Quantity during the specified Date\_Range. This 'safety\_target' Quantity is the max of Buffer.min\_on\_hand and the Quantity required to cover the next 'min\_time' of consumption in this Buffer.

For example, consider a Buffer with 'min\_on\_hand' set to "100", 'min\_time' is set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety\_target\_profile' will have three entries, first entry with quantity 100 until 4/1, second entry with quantity 200 starting on 4/1 and third entry with quantity 100 starting on 4/15. In the second period "200" will be maintained in order to cover the next "2 weeks" of out flows.

Properties: command=True Export-Only Field

**safety\_target (Date), safety\_target (Date\_Range, Logical) -- a Quantity field of model FIXED\_QUANTITY**

Safety target is a minimum Quantity this BASIC Buffer is required to maintain based on existing plan. Mathematically, 'safety\_target' Quantity is the max of Buffer.min\_on\_hand and the Quantity required to cover the next 'Buffer.min\_time' of consumption in this Buffer.

For example, consider a Buffer with 'min\_on\_hand' set to "100" and 'min\_time' set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety\_target' Quantity on date 4/10 will be 200.

If a Date is passed, this returns the 'safety\_target' Quantity required to be maintained as of that Date considering the existing plan.

If a Date\_Range is passed, this returns the smallest 'safety\_target' Quantity required to be maintained during that Date\_Range considering the existing plan.

If a Date\_Range and a Logical is passed, this returns the smallest 'safety\_target' Quantity required to be maintained during that Date\_Range considering the existing plan if the Logical is True. If the Logical is False, this returns the largest 'safety\_target' Quantity required to be maintained during that Date\_Range.

The Quantity is converted to the unit of this buffer.  
Properties: Export-Only Field

**cycle\_on\_hand\_profile, cycle\_on\_hand\_profile (Date\_Range) -- a**  
**List(Profile\_Quantity) field of model FIXED\_QUANTITY**

A profile or a list of changes to cycle 'on\_hand' (max(0.0, Buffer\_Plan.on\_hand - Buffer\_Plan.safety\_target)) Quantity's planned during the specified finite Date\_Range in this buffer.

Properties: command=True Export-Only Field

**cycle\_on\_hand (Date), cycle\_on\_hand (Date\_Range, Logical) -- a Quantity field of model FIXED\_QUANTITY**

Cycle 'on\_hand' (max(0.0, Buffer\_Plan.on\_hand - Buffer\_Plan.safety\_target)) Quantity planned in this buffer.

If a Date is passed, this returns the 'cycle\_on\_hand' Quantity planned as of that Date in this buffer.

If a Date\_Range is passed, this returns the smallest 'cycle\_on\_hand' Quantity planned during that Date\_Range in this buffer.

If a Date\_Range and a Logical is passed, this returns the smallest 'cycle\_on\_hand' Quantity planned during that Date\_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest 'max\_target' Quantity planned during that Date\_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

**excess\_on\_hand\_profile, excess\_on\_hand\_profile (Date\_Range) -- a**  
**List(Profile\_Quantity) field of model FIXED\_QUANTITY**

A profile or a list of changes to excess 'on\_hand' Quantity during the specified finite Date\_Range. This is the Buffer\_Plan.on\_hand' Quantity above allowable Buffer\_Plan.max\_target Quantity. Mathematically, 'excess\_on\_hand' is computed using formula, max(0.0, Buffer\_Plan.on\_hand - Buffer\_Plan.max\_target).  
Properties: command=True Export-Only Field

**excess\_on\_hand (Date), excess\_on\_hand (Date\_Range, Logical) -- a Quantity field of model FIXED\_QUANTITY**

Excess 'on\_hand' Quantity planned above the allowable Buffer.excess\_on\_hand (will be renamed to Buffer.max\_cycle\_quantity) Quantity in this buffer. Mathematically, 'excess\_on\_hand' is computed using formula, max(0.0, Buffer\_Plan.on\_hand - Buffer\_Plan.max\_target).

Extensions	MULTIPLE Extension
------------	--------------------

If a Date is passed, this returns the 'excess\_on\_hand' Quantity planned as of that Date.

If a Date\_Range is passed, this returns the smallest 'excess\_on\_hand' Quantity planned during that Date\_Range.

If a Date\_Range and a Logical is passed, this returns the smallest 'excess\_on\_hand' Quantity planned during that Date\_Range, if the Logical is True. If the Logical is False, this returns the largest 'excess\_on\_hand' Quantity planned during that Date\_Range.

The Quantity is converted to the 'unit' of this buffer.

Properties: Export-Only Field

`low_on_hand_profile, low_on_hand_profile (Date_Range) -- a`

`List(Profile_Quantity) field of model FIXED_QUANTITY`

A profile or a list of changes to the 'low\_on\_hand' Quantity during the specified Date\_Range. Mathematically, 'low\_on\_hand' is computed using following formula, (Buffer\_Plan.safety\_target(d) - Buffer\_Plan.on\_hand(d)).

Properties: command=True Export-Only Field

`low_on_hand (Date), low_on_hand (Date_Range, Logical) -- a Quantity field of model FIXED_QUANTITY`

It is the 'on\_hand' Quantity planned below the minimum 'safety\_target' Quantity. Mathematically, 'low\_on\_hand' is computed using following formula, (Buffer\_Plan.safety\_target - Buffer\_Plan.on\_hand).

If a Date is passed, this returns the 'low\_on\_hand' Quantity planned as of that Date.

If a Date\_Range is passed, this returns the smallest 'low\_on\_hand' Quantity planned during that Date\_Range.

If a Date\_Range and a Logical is passed, this returns the smallest 'low\_on\_hand' Quantity planned during that Date\_Range, if the Logical is True. If the Logical is False, this returns the largest 'low\_on\_hand' Quantity planned during that Date\_Range.

The Quantity is converted to the 'unit' of this buffer.

Properties: Export-Only Field

**MULTIPLE** -- a flow\_policy extension of model Buffer\_Plan

Extensions	MULTIPLE Extension
------------	--------------------

A MULTIPLE Buffer maintains a 'min\_time' safety time and a 'min\_on\_hand' safety quantity. See the corresponding 'flow\_policy' of the Buffer model for more details.

`max_target_profile, max_target_profile (Date_Range) -- a`

`List(Profile_Quantity) field of model MULTIPLE`

A profile or a list of changes to max allowable Quantity's during the specified finite Date\_Range. Up to 'max\_target' Quantity is allowed in this buffer. If the Buffer\_Plan.on\_hand' Quantity goes above 'Buffer\_Plan.max\_target' Quantity, RHYTHM will detect EXCESS\_ON\_HAND problem.

Properties: command=True Export-Only Field

`max_target (Date), max_target (Date_Range, Logical) -- a Quantity field of model MULTIPLE`

Maximum target Quantity allowed in this BASIC buffer. If the 'Buffer\_Plan.on\_hand' Quantity goes above 'Buffer\_Plan.max\_target' Quantity, RHYTHM will detect EXCESS\_ON\_HAND problem.

If a Date is passed, this returns the 'max\_target' Quantity allowed as of that Date in this buffer.

If a Date\_Range is passed, this returns the smallest 'max\_target' Quantity allowed during that Date\_Range in this buffer.

If a Date\_Range and a Logical is passed, this returns the smallest 'max\_target' Quantity allowed during that Date\_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest 'max\_target' Quantity allowed during that Date\_Range.

The Quantity is converted to the 'unit' of this buffer.

Properties: Export-Only Field

`safety_target_profile, safety_target_profile (Date_Range) -- a`

`List(Profile_Quantity) field of model MULTIPLE`

A profile or a list of changes to the 'safety\_target' Quantity during the specified Date\_Range. This 'safety\_target' Quantity is the max of Buffer.min\_on\_hand and the Quantity required to cover the next 'min\_time' of consumption in this Buffer.

For example, consider a Buffer with 'min\_on\_hand' is set to "100", 'min\_time' is set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety\_target\_profile' will have three entries, first entry with quantity 100 until 4/1, second entry with quantity 200 starting on 4/1 and third entry with quantity 100 starting on 4/15. In the second period "200" will be maintained in order to cover the next "2 weeks" of our flows.

Extensions	MULTIPLE Extension
------------	--------------------

Properties:    command=True    Export-Only Field

**safety\_target (Date), safety\_target (Date\_Range, Logical)    -- a Quantity field of model MULTIPLE**

Safety target is a minimum Quantity this BASIC Buffer is required to maintain based on existing plan. Mathematically, **safety\_target** Quantity is the max of **Buffer.min\_on\_hand** and the Quantity required to cover the next **Buffer.min\_time** of consumption in this Buffer.

For example, consider a Buffer with **min\_on\_hand** set to "100" and **min\_time** set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the **safety\_target** Quantity on date 4/10 will be 200.

If a Date is passed, this returns the **safety\_target** Quantity required to be maintained as of that Date considering the existing plan.

If a **Date\_Range** is passed, this returns the smallest **safety\_target** Quantity required to be maintained during that **Date\_Range** considering the existing plan.

If a **Date\_Range** and a Logical is passed, this returns the smallest **safety\_target** Quantity required to be maintained during that **Date\_Range** considering the existing plan if the Logical is True. If the Logical is False, this returns the largest **safety\_target** Quantity required to be maintained during that **Date\_Range**.

The Quantity is converted to the unit of this buffer.  
Properties:    Export-Only Field

**cycle\_on\_hand\_profile, cycle\_on\_hand\_profile (Date\_Range)    -- a List(Profile\_Quantity) field of model MULTIPLE**

A profile or a list of changes to **cycle\_on\_hand** (max(0.0, **Buffer\_Plan.on\_hand - Buffer\_Plan.safety\_target**)) Quantity's planned during the specified finite **Date\_Range** in this buffer.

Properties:    command=True    Export-Only Field

**cycle\_on\_hand (Date), cycle\_on\_hand (Date\_Range, Logical)    -- a Quantity field of model MULTIPLE**

Cycle **on\_hand** (max(0.0, **Buffer\_Plan.on\_hand - Buffer\_Plan.safety\_target**)) Quantity planned in this buffer.

If a Date is passed, this returns the **cycle\_on\_hand** Quantity planned as of that Date in this buffer.

Extensions	MULTIPLE Extension
------------	--------------------

If a **Date\_Range** is passed, this returns the smallest **cycle\_on\_hand** Quantity planned during that **Date\_Range** in this buffer.

If a **Date\_Range** and a Logical is passed, this returns the smallest **cycle\_on\_hand** Quantity planned during that **Date\_Range** in this buffer if the Logical is True. If the Logical is False, this returns the largest **max\_target** Quantity planned during that **Date\_Range**.

The Quantity is converted to the unit of this buffer.  
Properties:    Export-Only Field

**excess\_on\_hand\_profile, excess\_on\_hand\_profile (Date\_Range)    -- a List(Profile\_Quantity) field of model MULTIPLE**

A profile or a list of changes to **excess\_on\_hand** Quantity during the specified finite **Date\_Range**. This is the **Buffer\_Plan.on\_hand** Quantity above allowable **Buffer\_Plan.max\_target** Quantity. Mathematically, **excess\_on\_hand** is computed using formula, max(0.0, **Buffer\_Plan.on\_hand - Buffer\_Plan.max\_target**).  
Properties:    command=True    Export-Only Field

**excess\_on\_hand (Date), excess\_on\_hand (Date\_Range, Logical)    -- a Quantity field of model MULTIPLE**  
Excess **on\_hand** Quantity planned above the allowable **Buffer.excess\_on\_hand** (will be renamed to **Buffer.max\_cycle\_quantity**) Quantity in this buffer. Mathematically, **excess\_on\_hand** is computed using formula, max(0.0, **Buffer\_Plan.on\_hand - Buffer\_Plan.max\_target**).

If a Date is passed, this returns the **excess\_on\_hand** Quantity planned as of that Date.

If a **Date\_Range** is passed, this returns the smallest **excess\_on\_hand** Quantity planned during that **Date\_Range**.

If a **Date\_Range** and a Logical is passed, this returns the smallest **excess\_on\_hand** Quantity planned during that **Date\_Range**, if the Logical is True. If the Logical is False, this returns the largest **excess\_on\_hand** Quantity planned during that **Date\_Range**.

The Quantity is converted to the unit of this buffer.  
Properties:    Export-Only Field

**low\_on\_hand\_profile, low\_on\_hand\_profile (Date\_Range) -- a**  
**List(Profile\_Quantity) field of model MULTIPLE**

A profile or a list of changes to the low\_on\_hand Quantity during the specified Date\_Range. Mathematically, low\_on\_hand is computed using following formula.  
(Buffer\_Plan.safety\_target(d) - Buffer\_Plan.on\_hand(d)).

Properties: command=True Export-Only Field

**low\_on\_hand (Date), low\_on\_hand (Date\_Range, Logical) -- a Quantity field of model MULTIPLE**

It is the on\_hand Quantity planned below the minimum safety\_target Quantity. Mathematically, low\_on\_hand is computed using following formula,  
(Buffer\_Plan.safety\_target - Buffer\_Plan.on\_hand).

If a Date is passed, this returns the low\_on\_hand Quantity planned as of that Date.

If a Date\_Range is passed, this returns the smallest low\_on\_hand Quantity planned during that Date\_Range.

If a Date\_Range and a Logical is passed, this returns the smallest low\_on\_hand Quantity planned during that Date\_Range, if the Logical is True. If the Logical is False, this returns the largest low\_on\_hand Quantity planned during that Date\_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

#### MULTIPLE\_FILTER\_AND\_RANK -- a flow\_policy extension of model Buffer\_Plan

A MULTIPLE\_FILTER\_AND\_RANK Buffer maintains a min\_time safety time and a min\_on\_hand safety quantity. See the corresponding flow\_policy of the Buffer model for more details.

**max\_target\_profile, max\_target\_profile (Date\_Range) -- a**

**List(Profile\_Quantity) field of model MULTIPLE\_FILTER\_AND\_RANK**

A profile or a list of changes to max allowable Quantity's during the specified finite Date\_Range. Up to max\_target Quantity is allowed in this buffer. If the

Buffer\_Plan.on\_hand Quantity goes above Buffer\_Plan.max\_target Quantity, RHYTHM will detect EXCESS\_ON\_HAND problem.

Properties: command=True Export-Only Field

**max\_target (Date), max\_target (Date\_Range, Logical) -- a Quantity field of model MULTIPLE\_FILTER\_AND\_RANK**

Maximum target Quantity allowed in this BASIC buffer. If the Buffer\_Plan.on\_hand Quantity goes above Buffer\_Plan.max\_target Quantity, RHYTHM will detect EXCESS\_ON\_HAND problem.

If a Date is passed, this returns the max\_target Quantity allowed as of that Date in this buffer.

If a Date\_Range is passed, this returns the smallest max\_target Quantity allowed during that Date\_Range in this buffer.

If a Date\_Range and a Logical is passed, this returns the smallest max\_target Quantity allowed during that Date\_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest max\_target Quantity allowed during that Date\_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

**safety\_target\_profile, safety\_target\_profile (Date\_Range) -- a**

**List(Profile\_Quantity) field of model MULTIPLE\_FILTER\_AND\_RANK**

A profile or a list of changes to the safety\_target Quantity during the specified Date\_Range. This safety\_target Quantity is the max of Buffer.min\_on\_hand and the

Quantity required to cover the next min\_time of consumption in this Buffer.

For example, consider a Buffer with min\_on\_hand is set to "100", min\_time is set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the safety\_target\_profile will have three entries, first entry with quantity 100 until 4/1, second entry with quantity 200 starting on 4/1 and third entry with quantity 100 starting on 4/15. In the second period "200" will be maintained in order to cover the next "2 weeks" of out flows.

Properties: command=True Export-Only Field

**safety\_target (Date), safety\_target (Date\_Range, Logical) -- a Quantity field of model MULTIPLE\_FILTER\_AND\_RANK**

Safety target is a minimum Quantity this BASIC Buffer is required to maintain based on existing plan. Mathematically, safety\_target Quantity is the max of Buffer.min\_on\_hand and the Quantity required to cover the next Buffer.min\_time of consumption in this Buffer.

For example, consider a Buffer with min\_on\_hand set to "100" and min\_time set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the safety\_target Quantity on date 4/10 will be 200.

If a Date is passed, this returns the safety\_target Quantity required to be maintained as of that Date considering the existing plan.

If a Date\_Range is passed, this returns the smallest safety\_target Quantity required to be maintained during that Date\_Range considering the existing plan.

If a Date\_Range and a Logical is passed, this returns the smallest safety\_target Quantity required to be maintained during that Date\_Range considering the existing plan if the Logical is True. If the Logical is False, this returns the largest safety\_target Quantity required to be maintained during that Date\_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

cycle\_on\_hand\_profile, cycle\_on\_hand\_profile (Date\_Range) -- a

List(Profile\_Quantity) field of model MULTIPLE\_FILTER\_AND\_RANK

A profile or a list of changes to cycle\_on\_hand (max(0.0, Buffer\_Plan.on\_hand - Buffer\_Plan.safety\_target)) Quantity's planned during the specified finite Date\_Range in this buffer.

Properties: command=True Export-Only Field

cycle\_on\_hand (Date), cycle\_on\_hand (Date\_Range, Logical) -- a Quantity

field of model MULTIPLE\_FILTER\_AND\_RANK

Cycle\_on\_hand (max(0.0, Buffer\_Plan.on\_hand - Buffer\_Plan.safety\_target)) Quantity planned in this buffer.

If a Date is passed, this returns the cycle\_on\_hand Quantity planned as of that Date in this buffer.

If a Date\_Range is passed, this returns the smallest cycle\_on\_hand Quantity planned during that Date\_Range in this buffer.

If a Date\_Range and a Logical is passed, this returns the smallest cycle\_on\_hand Quantity planned during that Date\_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest max\_target Quantity planned during that Date\_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

excess\_on\_hand\_profile, excess\_on\_hand\_profile (Date\_Range) -- a

List(Profile\_Quantity) field of model MULTIPLE\_FILTER\_AND\_RANK

A profile or a list of changes to excess\_on\_hand Quantity during the specified finite Date\_Range. This is the Buffer\_Plan.on\_hand Quantity above allowable Buffer\_Plan.max\_target Quantity. Mathematically, excess\_on\_hand is computed using formula, max(0.0, Buffer\_Plan.on\_hand - Buffer\_Plan.max\_target).

Properties: command=True Export-Only Field

excess\_on\_hand (Date), excess\_on\_hand (Date\_Range, Logical) -- a Quantity

field of model MULTIPLE\_FILTER\_AND\_RANK

Excess\_on\_hand Quantity planned above the allowable Buffer.excess\_on\_hand will be renamed to Buffer.max\_cycle\_quantity Quantity in this buffer. Mathematically, excess\_on\_hand is computed using formula, max(0.0, Buffer\_Plan.on\_hand - Buffer\_Plan.max\_target).

If a Date is passed, this returns the excess\_on\_hand Quantity planned as of that Date.

If a Date\_Range is passed, this returns the smallest excess\_on\_hand Quantity planned during that Date\_Range.

If a Date\_Range and a Logical is passed, this returns the smallest excess\_on\_hand Quantity planned during that Date\_Range, if the Logical is True. If the Logical is False, this returns the largest excess\_on\_hand Quantity planned during that Date\_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

low\_on\_hand\_profile, low\_on\_hand\_profile (Date\_Range) -- a

List(Profile\_Quantity) field of model MULTIPLE\_FILTER\_AND\_RANK

A profile or a list of changes to the low\_on\_hand Quantity during the specified Date\_Range. Mathematically, low\_on\_hand is computed using following formula, (Buffer\_Plan.safety\_target(d) - Buffer\_Plan.on\_hand(d)).

Properties: command=True Export-Only Field

low\_on\_hand (Date), low\_on\_hand (Date\_Range, Logical) -- a Quantity field of model MULTIPLE\_FILTER\_AND\_RANK

It is the 'on\_hand' Quantity planned below the minimum 'safety\_target' Quantity. Mathematically, low\_on\_hand is computed using following formula, (Buffer\_Plan.safety\_target - Buffer\_Plan.on\_hand).

If a Date is passed, this returns the low\_on\_hand Quantity planned as of that Date.

If a Date\_Range is passed, this returns the smallest low\_on\_hand Quantity planned during that Date\_Range.

If a Date\_Range and a Logical is passed, this returns the smallest low\_on\_hand Quantity planned during that Date\_Range, if the Logical is True. If the Logical is False, this returns the largest low\_on\_hand Quantity planned during that Date\_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

FIXED\_QUANTITY\_FENCED -- a flow\_policy extension of model Buffer\_Plan

A FIXED\_QUANTITY\_FENCED Buffer maintains a 'min\_time' safety time and a 'min\_on\_hand' safety quantity. See the corresponding 'flow\_policy' of the Buffer model for more details.

max\_target\_profile, max\_target\_profile (Date\_Range) -- a list(Profile\_Quantity) field of model FIXED\_QUANTITY\_FENCED

A profile or a list of changes to max allowable Quantity's during the specified finite Date\_Range. Up to 'max\_target' Quantity is allowed in this buffer. If the Buffer\_Plan.on\_hand Quantity goes above 'Buffer\_Plan.max\_target' Quantity, RHYTHM will detect EXCESS\_ON\_HAND problem.

Properties: command=True Export-Only Field

max\_target (Date), max\_target (Date\_Range, Logical) -- a Quantity field of model FIXED\_QUANTITY\_FENCED

Maximum target Quantity allowed in this BASIC buffer. If the Buffer\_Plan.on\_hand Quantity goes above 'Buffer\_Plan.max\_target' Quantity, RHYTHM will detect EXCESS\_ON\_HAND problem.

If a Date is passed, this returns the 'max\_target' Quantity allowed as of that Date in this buffer.

If a Date\_Range is passed, this returns the smallest 'max\_target' Quantity allowed during that Date\_Range in this buffer.

If a Date\_Range and a Logical is passed, this returns the smallest 'max\_target' Quantity allowed during that Date\_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest 'max\_target' Quantity allowed during that Date\_Range.

The Quantity is converted to the unit of this buffer.

Properties: Export-Only Field

safety\_target\_profile, safety\_target\_profile (Date\_Range) -- a list(Profile\_Quantity) field of model FIXED\_QUANTITY\_FENCED

A profile or a list of changes to the 'safety\_target' Quantity during the specified Date\_Range. This 'safety\_target' Quantity is the max of Buffer.min\_on\_hand and the Quantity required to cover the next 'min\_time' of consumption in this Buffer.

For example, consider a Buffer with 'min\_on\_hand' is set to "100", 'min\_time' is set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety\_target\_profile' will have three entries, first entry with quantity 100 until 4/1, second entry with quantity 200 starting on 4/1 and third entry with quantity 100 starting on 4/15. In the second period "200" will be maintained in order to cover the next "2 weeks" of out flows.

Properties: command=True Export-Only Field

safety\_target (Date), safety\_target (Date\_Range, Logical) -- a Quantity field of model FIXED\_QUANTITY\_FENCED

Safety target is a minimum Quantity this BASIC Buffer is required to maintain based on existing plan. Mathematically, 'safety\_target' Quantity is the max of Buffer.min\_on\_hand and the Quantity required to cover the next 'Buffer.min\_time' of consumption in this Buffer.

For example, consider a Buffer with 'min\_on\_hand' set to "100" and 'min\_time' set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety\_target' Quantity on date 4/10 will be 200.

If a Date is passed, this returns the 'safety\_target' Quantity required to be maintained as of that Date considering the existing plan.

If a Date\_Range is passed, this returns the smallest 'safety\_target' Quantity required to be maintained during that Date\_Range considering the existing plan.

<i>Extensions</i>	<i>FIXED_QUANTITY_FENCED Extension</i>
-------------------	--

If a Date\_Range and a Logical is passed, this returns the smallest 'safety\_target' Quantity required to be maintained during that Date\_Range considering the existing plan if the Logical is True. If the Logical is False, this returns the largest 'safety\_target' Quantity required to be maintained during that Date\_Range.

The Quantity is converted to the 'unit' of this buffer.  
Properties:    Export-Only Field

cycle\_on\_hand\_profile, cycle\_on\_hand\_profile (Date\_Range)    -- *a*  
List(Profile\_Quantity)/*field of model* FIXED\_QUANTITY\_FENCED  
A profile or a list of changes to cycle 'on\_hand' (max(0.0, Buffer\_Plan.on\_hand - Buffer\_Plan.safety\_target)) Quantity's planned during the specified finite Date\_Range in this buffer.

Properties:    command=True    Export-Only Field

cycle\_on\_hand (Date), cycle\_on\_hand (Date\_Range, Logical)    -- *a* Quantity  
*field of model* FIXED\_QUANTITY\_FENCED  
Cycle 'on\_hand' (max(0.0, Buffer\_Plan.on\_hand - Buffer\_Plan.safety\_target)) Quantity planned in this buffer.

If a Date is passed, this returns the 'cycle\_on\_hand' Quantity planned as of that Date in this buffer.

If a Date\_Range is passed, this returns the smallest 'cycle\_on\_hand' Quantity planned during that Date\_Range in this buffer.

If a Date\_Range and a Logical is passed, this returns the smallest 'cycle\_on\_hand' Quantity planned during that Date\_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest 'max\_target' Quantity planned during that Date\_Range.

The Quantity is converted to the 'unit' of this buffer.  
Properties:    Export-Only Field

excess\_on\_hand\_profile, excess\_on\_hand\_profile (Date\_Range)    -- *a*  
List(Profile\_Quantity)/*field of model* FIXED\_QUANTITY\_FENCED  
A profile or a list of changes to excess 'on\_hand' Quantity during the specified finite Date\_Range. This is the 'Buffer\_Plan.on\_hand' Quantity above allowable Buffer\_Plan.max\_target Quantity. Mathematically, 'excess\_on\_hand' is computed using formula, max(0.0, Buffer\_Plan.on\_hand - Buffer\_Plan.max\_target).  
Properties:    command=True    Export-Only Field

<i>Extensions</i>	<i>FIXED_QUANTITY_FENCED Extension</i>
-------------------	--

excess\_on\_hand (Date), excess\_on\_hand (Date\_Range, Logical)    -- *a* Quantity  
*field of model* FIXED\_QUANTITY\_FENCED  
Excess 'on\_hand' Quantity planned above the allowable 'Buffer.excess\_on\_hand' will be renamed to Buffer.max\_cycle\_quantity' Quantity in this buffer. Mathematically, 'excess\_on\_hand' is computed using formula, max(0.0, Buffer\_Plan.on\_hand - Buffer\_Plan.max\_target).

If a Date is passed, this returns the 'excess\_on\_hand' Quantity planned as of that Date.  
If a Date\_Range is passed, this returns the smallest 'excess\_on\_hand' Quantity planned during that Date\_Range.

If a Date\_Range and a Logical is passed, this returns the smallest 'excess\_on\_hand' Quantity planned during that Date\_Range, if the Logical is True. If the Logical is False, this returns the largest 'excess\_on\_hand' Quantity planned during that Date\_Range.

The Quantity is converted to the 'unit' of this buffer.  
Properties:    Export-Only Field

low\_on\_hand\_profile, low\_on\_hand\_profile (Date\_Range)    -- *a*  
List(Profile\_Quantity)/*field of model* FIXED\_QUANTITY\_FENCED  
A profile or a list of changes to the 'low\_on\_hand' Quantity during the specified Date\_Range. Mathematically, 'low\_on\_hand' is computed using following formula, (Buffer\_Plan.safety\_target(d) - Buffer\_Plan.on\_hand(d)).  
Properties:    command=True    Export-Only Field

low\_on\_hand (Date), low\_on\_hand (Date\_Range, Logical)    -- *a* Quantity/*field of model* FIXED\_QUANTITY\_FENCED  
It is the 'on\_hand' Quantity planned below the minimum 'safety\_target' Quantity. Mathematically, 'low\_on\_hand' is computed using following formula, (Buffer\_Plan.safety\_target - Buffer\_Plan.on\_hand).

If a Date is passed, this returns the 'low\_on\_hand' Quantity planned as of that Date.  
If a Date\_Range is passed, this returns the smallest 'low\_on\_hand' Quantity planned during that Date\_Range.

If a `Date_Range` and a Logical is passed, this returns the smallest 'low\_on\_hand' Quantity planned during that `Date_Range`, if the Logical is True. If the Logical is False, this returns the largest 'low\_on\_hand' Quantity planned during that `Date_Range`.

The Quantity is converted to the 'unit' of this buffer:

Properties: Export-Only Field

#### FIXED\_TIME -- a flow\_policy extension of model Buffer\_Plan

A `FIXED_TIME` Buffer maintains a 'min\_time' safety time and a 'min\_on\_hand' safety quantity. See the corresponding 'flow\_policy' of the Buffer model for more details.

`max_target_profile`, `max_target_profile (Date_Range)` -- a

`List(Profile_Quantity) field of model FIXED_TIME`

A profile or a list of changes to max allowable Quantity's during the specified finite

`Date_Range`. Up to `max_target` Quantity is allowed in this buffer. If the

Buffer\_Plan.on\_hand' Quantity goes above Buffer\_Plan.max\_target' Quantity,

RHYTHM will detect `EXCESS_ON_HAND` problem.

Properties: command=True Export-Only Field

`max_target (Date)`, `max_target (Date_Range, Logical)` -- a Quantity field of  
model `FIXED_TIME`

Maximum target Quantity allowed in this BASIC buffer. If the Buffer\_Plan.on\_hand' Quantity goes above Buffer\_Plan.max\_target' Quantity, RHYTHM will detect `EXCESS_ON_HAND` problem.

If a Date is passed, this returns the 'max\_target' Quantity allowed as of that Date in this buffer.

If a `Date_Range` is passed, this returns the smallest 'max\_target' Quantity allowed during that `Date_Range` in this buffer.

If a `Date_Range` and a Logical is passed, this returns the smallest 'max\_target' Quantity allowed during that `Date_Range` in this buffer if the Logical is True. If the Logical is False, this returns the largest 'max\_target' Quantity allowed during that `Date_Range`.

The Quantity is converted to the 'unit' of this buffer:

Properties: Export-Only Field

`safety_target_profile`, `safety_target_profile (Date_Range)` -- a

`List(Profile_Quantity) field of model FIXED_TIME`

A profile or a list of changes to the 'safety\_target' Quantity during the specified `Date_Range`. This 'safety\_target' Quantity is the max of Buffer.min\_on\_hand and the Quantity required to cover the next 'min\_time' of consumption in this Buffer.

For example, consider a Buffer with 'min\_on\_hand' is set to "100", 'min\_time' is set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety\_target\_profile' will have three entries, first entry with quantity 100 until 4/1, second entry with quantity 200 starting on 4/1 and third entry with quantity 100 starting on 4/15. In the second period "200" will be maintained in order to cover the next "2 weeks" of out flows.

Properties: command=True Export-Only Field

`safety_target (Date)`, `safety_target (Date_Range, Logical)` -- a Quantity field of  
model `FIXED_TIME`

Safety target is a minimum Quantity this BASIC Buffer is required to maintain based on existing plan. Mathematically, 'safety\_target' Quantity is the max of

Buffer.min\_on\_hand and the Quantity required to cover the next 'Buffer.min\_time' of consumption in this Buffer.

For example, consider a Buffer with 'min\_on\_hand' set to "100" and 'min\_time' set to "2 weeks". If there is currently just one out flow planned from the Buffer, and it is for 200 units on 4/15, then the 'safety\_target' Quantity on date 4/10 will be 200.

If a Date is passed, this returns the 'safety\_target' Quantity required to be maintained as of that Date considering the existing plan.

If a `Date_Range` is passed, this returns the smallest 'safety\_target' Quantity required to be maintained during that `Date_Range` considering the existing plan.

If a `Date_Range` and a Logical is passed, this returns the smallest 'safety\_target' Quantity required to be maintained during that `Date_Range` considering the existing plan if the Logical is True. If the Logical is False, this returns the largest 'safety\_target' Quantity required to be maintained during that `Date_Range`.

The Quantity is converted to the 'unit' of this buffer:

Properties: Export-Only Field



Extensions	FIXED_TIME Extension
------------	----------------------

cycle\_on\_hand\_profile, cycle\_on\_hand\_profile (Date\_Range) -- a  
 List(Profile\_Quantity) *field of model FIXED\_TIME*  
 A profile or a list of changes to cycle 'on\_hand' (max(0,0, Buffer\_Plan.on\_hand - Buffer\_Plan.safety\_target)) Quantity's planned during the specified finite Date\_Range in this buffer.  
 Properties:    command=True    Export-Only Field

cycle\_on\_hand (Date), cycle\_on\_hand (Date\_Range, Logical) -- a Quantity  
*field of model FIXED\_TIME*  
 Cycle 'on\_hand' (max(0,0, Buffer\_Plan.on\_hand - Buffer\_Plan.safety\_target)) Quantity planned in this buffer.  
 If a Date is passed, this returns the 'cycle\_on\_hand' Quantity planned as of that Date in this buffer.  
 If a Date\_Range is passed, this returns the smallest 'cycle\_on\_hand' Quantity planned during that Date\_Range in this buffer.  
 If a Date\_Range and a Logical is passed, this returns the smallest 'cycle\_on\_hand' Quantity planned during that Date\_Range in this buffer if the Logical is True. If the Logical is False, this returns the largest 'max\_target' Quantity planned during that Date\_Range.  
 The Quantity is converted to the 'unit' of this buffer.  
 Properties:    Export-Only Field

excess\_on\_hand\_profile, excess\_on\_hand\_profile (Date\_Range) -- a  
 List(Profile\_Quantity) *field of model FIXED\_TIME*  
 A profile or a list of changes to excess 'on\_hand' Quantity during the specified finite Date\_Range. This is the 'Buffer\_Plan.on\_hand' Quantity above allowable Buffer\_Plan.max\_target Quantity. Mathematically, 'excess\_on\_hand' is computed using formula, max(0,0, Buffer\_Plan.on\_hand - Buffer\_Plan.max\_target).  
 Properties:    command=True    Export-Only Field

excess\_on\_hand (Date), excess\_on\_hand (Date\_Range, Logical) -- a Quantity  
*field of model FIXED\_TIME*  
 Excess 'on\_hand' Quantity planned above the allowable 'Buffer.excess\_on\_hand' (will be renamed to 'Buffer.max\_cycle\_quantity') Quantity in this buffer. Mathematically, 'excess\_on\_hand' is computed using formula, max(0,0, Buffer\_Plan.on\_hand - Buffer\_Plan.max\_target).

Extensions	FIXED_TIME Extension
------------	----------------------

If a Date is passed, this returns the 'excess\_on\_hand' Quantity planned as of that Date.  
 If a Date\_Range is passed, this returns the smallest 'excess\_on\_hand' Quantity planned during that Date\_Range.  
 If a Date\_Range and a Logical is passed, this returns the smallest 'excess\_on\_hand' Quantity planned during that Date\_Range, if the Logical is True. If the Logical is False, this returns the largest 'excess\_on\_hand' Quantity planned during that Date\_Range.  
 The Quantity is converted to the 'unit' of this buffer.  
 Properties:    Export-Only Field

low\_on\_hand\_profile, low\_on\_hand\_profile (Date\_Range) -- a  
 List(Profile\_Quantity) *field of model FIXED\_TIME*  
 A profile or a list of changes to the 'low\_on\_hand' Quantity during the specified Date\_Range. Mathematically, 'low\_on\_hand' is computed using following formula, (Buffer\_Plan.safety\_target(d) - Buffer\_Plan.on\_hand(d)).  
 Properties:    command=True    Export-Only Field

low\_on\_hand (Date), low\_on\_hand (Date\_Range, Logical) -- a Quantity *field of model FIXED\_TIME*  
 It is the 'on\_hand' Quantity planned below the minimum 'safety\_target' Quantity. Mathematically, 'low\_on\_hand' is computed using following formula, (Buffer\_Plan.safety\_target - Buffer\_Plan.on\_hand).  
 If a Date is passed, this returns the 'low\_on\_hand' Quantity planned as of that Date.  
 If a Date\_Range is passed, this returns the smallest 'low\_on\_hand' Quantity planned during that Date\_Range.  
 If a Date\_Range and a Logical is passed, this returns the smallest 'low\_on\_hand' Quantity planned during that Date\_Range, if the Logical is True. If the Logical is False, this returns the largest 'low\_on\_hand' Quantity planned during that Date\_Range.  
 The Quantity is converted to the 'unit' of this buffer.  
 Properties:    Export-Only Field

<i>Extensions</i>	<i>Forecast Extensions</i>
-------------------	----------------------------

### 10.19 Forecast Extensions

#### 10.19.1 level extensions of model Forecast

##### INDIVIDUAL -- a level extension of model Forecast

An INDIVIDUAL Forecast is for an individual Product. The 'product' defines how this Forecast is converted into Requests upon this Site.

The INDIVIDUAL model has these submodels :  
ATP\_Entry.

The INDIVIDUAL model has fields that references these models :  
Product, ATP\_Entry.

**product** - - a Product field of model INDIVIDUAL

The Product for which demand this forecasts.  
Properties: standard=True Export-Only Field

**atp\_entries** - - a list of ATP\_Entry submodels of model INDIVIDUAL  
In each Forecast, there is one ATP\_Entry for each Date\_Range in the Seller\_Plan's list of Date\_Ranges, 'atp\_horizon'. Each ATP\_Entry has the Date\_Range for which it maintains allocations, an 'allocated' value that it can promise, a 'consumed' value that maintains the amount already promised, and an 'atp' value that is still available to promise.

##### GROUP -- a level extension of model Forecast

A GROUP Forecast is for a Product\_Group, an aggregation of several Products' Forecasts. The 'product\_group' defines how this Forecast is propagated down to the Product\_Group's 'sub\_groups' and 'products'.

The GROUP model has fields that references these models :  
Product\_Group.

**product\_group** - - a Product\_Group field of model GROUP

The Product for which demand this forecasts.  
Properties: standard=True Export-Only Field

**sub\_forecasts** - - a List(Forecast) field of model GROUP  
The Forecasts for the 'sub\_groups' and 'products' of this Forecast's 'product\_group'.  
Properties: standard=True Export-Only Field

<i>Extensions</i>	<i>GROUP Extension</i>
-------------------	------------------------

**active\_sub\_forecasts** - - a List(Forecast) field of model GROUP  
The Forecasts for the 'sub\_groups' and active 'products' of this Forecast's 'product\_group'.  
Properties: standard=True Export-Only Field

**active\_leaf\_product\_forecasts** - - a List(Forecast) field of model GROUP  
The Forecasts for the active 'products' of this Forecast's 'product\_group'. Returns all active descendants of this GROUP Forecast that are INDIVIDUAL forecasts. In other terms, these are the active "leaf" Forecasts of this product sub-hierarchy.  
Properties: standard=True Export-Only Field

**use\_std\_split** - - a Logical field of model GROUP  
This field is obsolete. You must now set the 'use\_std\_split' field in the Product\_Group. Note that setting the field at the Product\_Group is not exactly equivalent. The split now applies to all Forecasts of the Product\_Group.  
Properties: obsolete=True Export-Only Field

## 10.20 Forecast\_Entry Extensions

10.20.1 forecast\_policy extensions of model Forecast\_Entry

10.20.2 allocation\_policy extensions of model Forecast\_Entry

MEMBER\_RANK -- *a allocation\_policy extension of model Forecast\_Entry*

The MEMBER\_RANK allocation\_policy distributes the allocations to the 'members' in the order of their 'rank'. For Sellers with equal 'rank', it distributes to each proportional to the quantity 'committed' to by that Seller. A portion of the allocation can be retained for use at the discretion of this Seller. Any leftovers due to lot sizing or explicit adjustments will be available on a first-come-first-served (FCFS) basis.

**pooled\_allocated** -- *a Quantity field of model MEMBER\_RANK*  
collective allocated quantity for members whose 'lock\_allocated' field is set to true  
Default: 0

**pooled\_accepted** -- *a Quantity field of model MEMBER\_RANK*  
collective accepted quantity for members whose 'lock\_allocated' field is set to true  
Default: 0  
Properties: Export-Only Field

**pooled\_allocated\_available** -- *a Quantity field of model MEMBER\_RANK*  
collective allocated\_available quantity for members whose 'lock\_allocated' field is set to true  
Properties: Export-Only Field

**pooled\_available** -- *a Quantity field of model MEMBER\_RANK*  
collective accepted\_available quantity for members whose 'lock\_allocated' field is set to true  
Properties: Export-Only Field

## 10.21 Site\_Plan Extensions

10.21.1 role extensions of model Site\_Plan

LINK -- *a role extension of model Site\_Plan*

A LINK Site is considered to be a "link" of this supply "chain".

The LINK model has these submodels:  
**Buffer\_Plan, Resource\_Plan, Operation\_Plan, Operation\_State, Request, Promise, Acceptance.**

The LINK model has fields that references these models:  
**Buffer\_Plan, Resource\_Plan, Operation\_Plan, Operation\_State, Request, Promise, Acceptance.**

**buffer\_plans** -- *a list of Buffer\_Plan submodels of model LINK*  
The plans for each Buffer defined in the site\_plan site. Each buffer in each Site will have exactly one Buffer\_Plan per Site\_Plan.

**resource\_plans** -- *a list of Resource\_Plan submodels of model LINK*  
The plans for each Resource defined in this site\_plan's site. Each resource in each Site will have one Resource\_Plan per Site\_Plan.

**operation\_plans** -- *a list of Operation\_Plan submodels of model LINK*  
The plans for each Operation defined in this site\_plan's site. Each operation in each Site could have multiple Operation\_Plans per Site\_Plan.

**operation\_states** -- *a list of Operation\_State submodels of model LINK*  
State reports which will be assigned to Operation\_Plans.

**requests** -- *a list of Request submodels of model LINK*  
The Requests that have been made by other Sites for Items supplied by this Site. This is the raw demand on this Site.

**promises** -- *a list of Promise submodels of model LINK*  
The promises that have been made by this Site to supply Items to other Sites. This is the demand this Site has agreed to fulfill. By some definitions, this is the master production schedule.

acceptances - - a list of Acceptance submodels of model LINK

The acceptances that have been made by the other Site in response to the promises from this site

plan\_to\_satisfy\_unanswered\_requests - - a Void field of model LINK

This command creates plans to satisfy all the Requests that have been made upon this Site\_Plan that have not yet been "answered". It does this by simply calling plan\_to\_satisfy' on each of its "unanswered" requests'.

Alternatively, plan\_to\_satisfy\_all\_promises' can create plans such that all Promises would be satisfied; or plan\_to\_satisfy\_all\_requests' to create plans such that all Requests upon this Site\_Plan are satisfied.

Properties: command=True Export-Only Field

plan\_to\_satisfy\_queued\_requests - - a Void field of model LINK

This command creates plans to satisfy all the Requests that have been made upon this Site\_Plan that have been "queued" for later consideration. It does this by simply calling plan\_to\_satisfy' on each of its "queued" requests'.

Alternatively, plan\_to\_satisfy\_all\_promises' can create plans such that all Promises would be satisfied; or plan\_to\_satisfy\_all\_requests' to create plans such that all Requests upon this Site\_Plan are satisfied.

Properties: command=True Export-Only Field

plan\_to\_satisfy\_all\_requests - - a Void field of model LINK

This command creates plans to satisfy all the Requests that have been made upon this Site\_Plan. It does this by simply calling plan\_to\_satisfy' on each of its requests'. Thus, it is equivalent to requests\_for\_each(#plan\_to\_satisfy)'.

Alternatively, plan\_to\_satisfy\_all\_promises' can create plans such that all Promises would be satisfied; or plan\_to\_satisfy\_unanswered\_requests' to create plans such that Requests that have not yet been "answered" are satisfied.

Properties: command=True Export-Only Field

plan\_to\_satisfy\_all\_promises - - a Void field of model LINK

This command creates plans to satisfy all the Promises that have been made by this Site\_Plan. It does this by simply calling plan\_to\_satisfy' on each of its promises'. Thus, it is equivalent to promises\_for\_each(#plan\_to\_satisfy)'.

Alternatively, plan\_to\_satisfy\_all\_requests' can create plans such that all Requests would be satisfied; or plan\_to\_satisfy\_unanswered\_requests' to create plans such that Requests that have not yet been "answered" are satisfied.

Properties: command=True Export-Only Field

promise\_as\_planned - - a Void field of model LINK

This command sets this Site\_Plan's promises' to promise what has been planned for each. It does this by simply calling promise\_as\_planned' on each of its promises'. Thus, it is equivalent to promises\_for\_each(#promise\_as\_planned)'.

Properties: command=True Export-Only Field

supply\_requests - - a List(Request) field of model LINK

The Requests that have been made by this Site for Items supplied by other Sites.

Properties: Export-Only Field

supply\_promises - - a List(Promise) field of model LINK

The Promises that have been made by other Sites to supply Items to this Site.

Properties: Export-Only Field

supply\_item\_requests, supply\_item\_requests (Date\_Range) , supply\_item\_requests (Item) , supply\_item\_requests (Item, Date\_Range) - - a List(Item\_Request) field of model LINK

The Requests that have been made by this Site for Items supplied by other Sites.

Properties: Export-Only Field

item\_demand (List(Item), Date\_Range) - - a Quantity field of model LINK

Returns the sum of demand for a List of Items during a Date\_Range. The Quantity will be unitless as the sum of the number of units of each Item.

Properties: Export-Only Field

problems, problems (Date\_Range) , problems (Problem\_Category) , problems (Problem\_Category, Date\_Range) - - a List(Problem) field of model LINK

The Problems detected with this Site\_Plan. If passed a Problem\_Category, only the Problems with that category' are returned. If passed a Date\_Range, only the Problems whose dates' overlap are returned.

Note that you can pass in one of the special Problem\_Category's to get Problems in larger groups. For example, the OPERATION Problem\_Category will give all Problems in Operation-related Problem categories. Similarly for RESOURCE, BUFFER, and even ALL.

Properties: Export-Only Field

**problem\_categories** -- *a list(Problem\_Category) field of model LINK*

For each different 'category' of Problem in 'problems', a Problem\_Category is added to this list. It gives you the name of the 'category' (in various forms) plus a list of just the Problems of that 'category'. These sublists are often much easier to deal with than the full 'problems' list.

Note that this List will not include special Problem\_Categories, such as OPERATION, RESOURCE, BUFFER, or ALL.  
Properties: Export-Only Field

#### SUPPLIER -- *a role extension of model Site\_Plan*

A SUPPLIER Site is not planned or modeled in detail; rather it represents the Items that can be procured from a supplier by LINK Sites, the Requests for those Items, and the Promises received from the supplier.

The SUPPLIER model has these submodels :  
Request, Promise, Acceptance.

The SUPPLIER model has fields that references these models :  
Request, Promise, Acceptance.

**requests** -- *a list of Request submodels of model SUPPLIER*

The Requests that have been made by other Sites for Items supplied by this Site. This is the raw demand on this Site.

**promises** -- *a list of Promise submodels of model SUPPLIER*

The promises that have been made by this Site to supply Items to other Sites. This is the demand this Site has agreed to fulfill.

**acceptances** -- *a list of Acceptance submodels of model SUPPLIER*

The acceptances that have been made by the other Site in response to the promises from this site

#### CUSTOMER -- *a role extension of model Site\_Plan*

A CUSTOMER Site is not planned or modeled in detail; rather, it is simply a destination for deliveries and source of Requests.

## 10.22 Problem Extensions

### 10.22.1 category extensions of model Problem

#### REQUEST\_NOT\_PLANNED -- *a category extension of model Problem*

A REQUEST\_NOT\_PLANNED Problem indicates that the 'item\_request' was issued after the 'item\_promise' was offered -- and -- the 'delivery\_plan' has not been planned to satisfy the 'item\_request'; either there is no 'delivery\_plan', or the 'delivery\_plan' is for the older 'item\_promise'. This Problem will not occur if the 'delivery\_request' has an infinite 'due' Date, the 'item\_request' is for a zero 'quantity', or the 'item\_promise' was offered after the 'item\_request' was issued.

To resolve this Problem, either eliminate or zero out the 'item\_request' (unlikely), 'offer' a Promise, or 'plan\_to\_satisfy' the 'item\_request' which will create and/or replan the 'delivery\_plan' to meet the 'item\_request'.

The typical series of events begins with the receipt of a newly issued Request. Its 'Item\_Requests' will each have this REQUEST\_NOT\_PLANNED Problem. The planner either uses ATP to Promise them directly (eliminating this problem, replacing it with PROMISE\_NOT\_PLANNED), or the planner does 'plan\_to\_satisfy' (eliminating this Problem, replacing it with PROMISE\_NOT\_OFFERED).

The REQUEST\_QUEUED Problem is a variation of this Problem -- both are resolved the same way, but a REQUEST\_QUEUED Request has been considered before, whereas a REQUEST\_NOT\_PLANNED Request is new.

This is in the REQUEST\_Problem\_Set category, as well as REQUEST\_NOT\_PLANNED.

The REQUEST\_NOT\_PLANNED model has fields that references these models :  
Item\_Request, Item\_Promise, Item\_Acceptance, Delivery\_Request,  
Delivery\_Promise, Delivery\_Acceptance, Operation\_Plan.

**item\_request** -- *a Item\_Request field of model REQUEST\_NOT\_PLANNED*  
The Item\_Request that has this REQUEST\_NOT\_PLANNED Problem. This 'item\_request.delivery\_plan' either is nonexistent or is planned for this 'item\_request.item\_promise', which is older than this Item\_Request.  
Properties: Export-Only Field

**item\_promise** -- *a Item\_Promise field of model REQUEST\_NOT\_PLANNED*  
The Item\_Promise for the 'item\_request'.

Properties: Export-Only Field

item\_acceptance -- *a* Item\_Acceptance field of model REQUEST\_NOT\_PLANNED

The Item\_Acceptance for the 'item\_request'.  
Properties: Export-Only Field

delivery\_request -- *a* Delivery\_Request field of model REQUEST\_NOT\_PLANNED

This is simply shorthand for 'item\_request.owner'.  
Properties: Export-Only Field

delivery\_promise -- *a* Delivery\_Promise field of model REQUEST\_NOT\_PLANNED

This is simply shorthand for 'item\_promise.owner'.  
Properties: Export-Only Field

delivery\_acceptance -- *a* Delivery\_Acceptance field of model REQUEST\_NOT\_PLANNED

This is simply shorthand for 'item\_acceptance.owner'.  
Properties: Export-Only Field

request -- *a* Request field of model REQUEST\_NOT\_PLANNED

This is simply shorthand for 'item\_request.delivery\_request.owner'.  
Properties: Export-Only Field

promise -- *a* Promise field of model REQUEST\_NOT\_PLANNED

This is simply shorthand for 'item\_promise.delivery\_promise.owner'.  
Properties: Export-Only Field

acceptance -- *a* Acceptance field of model REQUEST\_NOT\_PLANNED

This is simply shorthand for 'item\_acceptance.delivery\_acceptance.owner'.  
Properties: Export-Only Field

delivery\_plan -- *a* Operation\_Plan field of model REQUEST\_NOT\_PLANNED

This is simply shorthand for 'item\_request.delivery\_plan'.  
Properties: Export-Only Field

REQUEST\_PLANNED\_LATE -- *a* category extension of model Problem

A REQUEST\_PLANNED\_LATE Problem indicates that the 'delivery\_plan' has been planned to satisfy the 'item\_request' but is in fact planned later than the 'delivery\_request's 'due' Dates. This Problem will not occur if the 'delivery\_request' has an infinite 'due' Date.

To resolve this Problem, either set the 'delivery\_request.due' to be later (unlikely), or adjust the 'delivery\_plan' such that its end Date is within 'due'.

This is in both the REQUEST and REQUEST\_PLAN Problem\_Sets categories, as well as REQUEST\_PLANNED\_LATE.

The REQUEST\_PLANNED\_LATE model has fields that references these models : Item\_Request, Item\_Promise, Item\_Acceptance, Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance, Operation\_Plan.

item\_request -- *a* Item\_Request field of model REQUEST\_PLANNED\_LATE

The Item\_Request that has this REQUEST\_PLANNED\_LATE Problem. This 'item\_request.delivery\_plan' is planned to end later than this 'item\_request.owner.due' (the due Date of this Item\_Request).  
Properties: Export-Only Field

item\_promise -- *a* Item\_Promise field of model REQUEST\_PLANNED\_LATE

The Item\_Promise for the 'item\_request'.  
Properties: Export-Only Field

item\_acceptance -- *a* Item\_Acceptance field of model REQUEST\_PLANNED\_LATE

The Item\_Acceptance for the 'item\_request'.  
Properties: Export-Only Field

delivery\_request -- *a* Delivery\_Request field of model REQUEST\_PLANNED\_LATE

This is simply shorthand for 'item\_request.owner'. The 'delivery\_plan' is planned to end later than this 'delivery\_request.due'.  
Properties: Export-Only Field

delivery\_promise -- *a* Delivery\_Promise field of model REQUEST\_PLANNED\_LATE

This is simply shorthand for 'item\_promise.owner'.  
Properties: Export-Only Field

Extensions	REQUEST_PLANNED_EARLY Extension
------------	---------------------------------

delivery\_acceptance - - a Delivery\_Acceptance field of model REQUEST\_PLANNED\_LATE  
This is simply shorthand for item\_acceptance.owner.  
Properties: Export-Only Field

request - - a Request field of model REQUEST\_PLANNED\_LATE  
This is simply shorthand for item\_request.delivery\_request.owner.  
Properties: Export-Only Field

promise - - a Promise field of model REQUEST\_PLANNED\_LATE  
This is simply shorthand for item\_promise.delivery\_promise.owner.  
Properties: Export-Only Field

acceptance - - a Acceptance field of model REQUEST\_PLANNED\_LATE  
This is simply shorthand for item\_acceptance.delivery\_acceptance.owner.  
Properties: Export-Only Field

delivery\_plan - - a Operation\_Plan field of model REQUEST\_PLANNED\_LATE  
This is simply shorthand for item\_request.delivery\_plan. This delivery\_plan is planned to end later than delivery\_request.due.  
Properties: Export-Only Field

**REQUEST\_PLANNED\_EARLY - - a category extension of model Problem**

A REQUEST\_PLANNED\_EARLY Problem indicates that the delivery\_plan has been planned to satisfy the delivery\_request but is in fact planned earlier than the delivery\_request's due Dates. This Problem will not occur if the delivery\_request has an infinite due Date.

To resolve this Problem, either set the delivery\_request.due to be earlier (unlikely), or adjust the delivery\_plan such that its end Date is within due.

This is in both the REQUEST and REQUEST\_PLAN Problem categories, as well as REQUEST\_PLANNED\_EARLY.

The REQUEST\_PLANNED\_EARLY model has fields that references these models :  
Item\_Request, Item\_Promise, Item\_Acceptance, Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance, Operation\_Plan.

Extensions	REQUEST_PLANNED_EARLY Extension
------------	---------------------------------

item\_request - - a Item\_Request field of model REQUEST\_PLANNED\_EARLY  
The Item\_Request that has this REQUEST\_PLANNED\_EARLY Problem. This item\_request.delivery\_plan is planned to end earlier than this item\_request.owner.due (the due Date of this Item\_Request).  
Properties: Export-Only Field

item\_promise - - a Item\_Promise field of model REQUEST\_PLANNED\_EARLY  
The Item\_Promise for the item\_request.  
Properties: Export-Only Field

item\_acceptance - - a Item\_Acceptance field of model REQUEST\_PLANNED\_EARLY  
The Item\_Acceptance for the item\_request.  
Properties: Export-Only Field

delivery\_request - - a Delivery\_Request field of model REQUEST\_PLANNED\_EARLY  
This is simply shorthand for item\_request.owner. The delivery\_plan is planned to end earlier than this delivery\_request.due.  
Properties: Export-Only Field

delivery\_promise - - a Delivery\_Promise field of model REQUEST\_PLANNED\_EARLY  
This is simply shorthand for item\_promise.owner.  
Properties: Export-Only Field

delivery\_acceptance - - a Delivery\_Acceptance field of model REQUEST\_PLANNED\_EARLY  
This is simply shorthand for item\_acceptance.owner.  
Properties: Export-Only Field

request - - a Request field of model REQUEST\_PLANNED\_EARLY  
This is simply shorthand for item\_request.delivery\_request.owner.  
Properties: Export-Only Field

promise - - a Promise field of model REQUEST\_PLANNED\_EARLY  
This is simply shorthand for item\_promise.delivery\_promise.owner.  
Properties: Export-Only Field

acceptance - - a Acceptance field of model REQUEST\_PLANNED\_EARLY  
This is simply shorthand for item\_acceptance.delivery\_acceptance.owner.

<i>Extensions</i>	<i>REQUEST_PLANNED_SHORT Extension</i>
-------------------	--

Properties:    Export-Only Field

**delivery\_plan** - - *a Operation\_Plan field of model REQUEST\_PLANNED\_EARLY*

This is simply shorthand for 'item\_request.delivery\_plan'. This 'delivery\_plan' is planned to end earlier than 'delivery\_request.due'.

Properties:    Export-Only Field

**REQUEST\_PLANNED\_SHORT** - - *a category extension of model Problem*

A REQUEST\_PLANNED\_SHORT Problem indicates that the 'delivery\_plan' has been planned to satisfy the 'item\_request' but is in fact planned for less 'quantity' than requested. This Problem will not occur if the 'item\_request' has an infinite 'quantity' range.

To resolve this Problem, either reduce 'item\_request.quantity' (unlikely), or increase the Quantity planned by 'item\_request.delivery\_plan' to be within the requested Quantity\_Range.

This is in both the REQUEST and REQUEST\_PLAN Problem\_Set categories, as well as REQUEST\_PLANNED\_SHORT.

The REQUEST\_PLANNED\_SHORT model has fields that references these models :  
**Item\_Request, Item\_Promise, Item\_Acceptance, Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance, Operation\_Plan.**

**item\_request** - - *a Item\_Request field of model REQUEST\_PLANNED\_SHORT*  
The Item\_Request that has this REQUEST\_PLANNED\_SHORT Problem. This 'item\_request.delivery\_plan' is planned to deliver less 'quantity' than this 'item\_request.quantity' range.  
Properties:    Export-Only Field

**item\_promise** - - *a Item\_Promise field of model REQUEST\_PLANNED\_SHORT*  
The Item\_Promise for the 'item\_request'.  
Properties:    Export-Only Field

**item\_acceptance** - - *a Item\_Acceptance field of model REQUEST\_PLANNED\_SHORT*  
The Item\_Acceptance for the 'item\_request'.  
Properties:    Export-Only Field

<i>Extensions</i>	<i>REQUEST_PLANNED_EXCESS Extension</i>
-------------------	---

**delivery\_request** - - *a Delivery\_Request field of model REQUEST\_PLANNED\_SHORT*

This is simply shorthand for 'item\_request.owner'.  
Properties:    Export-Only Field

**delivery\_promise** - - *a Delivery\_Promise field of model REQUEST\_PLANNED\_SHORT*  
This is simply shorthand for 'item\_promise.owner'.  
Properties:    Export-Only Field

**delivery\_acceptance** - - *a Delivery\_Acceptance field of model REQUEST\_PLANNED\_SHORT*

This is simply shorthand for 'item\_acceptance.owner'.  
Properties:    Export-Only Field

**request** - - *a Request field of model REQUEST\_PLANNED\_SHORT*  
This is simply shorthand for 'item\_request.delivery\_request.owner'.  
Properties:    Export-Only Field

**promise** - - *a Promise field of model REQUEST\_PLANNED\_SHORT*  
This is simply shorthand for 'item\_promise.delivery\_promise.owner'.  
Properties:    Export-Only Field

**acceptance** - - *a Acceptance field of model REQUEST\_PLANNED\_SHORT*  
This is simply shorthand for 'item\_acceptance.delivery\_acceptance.owner'.  
Properties:    Export-Only Field

**delivery\_plan** - - *a Operation\_Plan field of model REQUEST\_PLANNED\_SHORT*  
This is simply shorthand for 'item\_request.delivery\_plan'. This 'delivery\_plan' is planned to deliver less than 'item\_request.quantity'.  
Properties:    Export-Only Field

**REQUEST\_PLANNED\_EXCESS** - - *a category extension of model Problem*

A REQUEST\_PLANNED\_EXCESS Problem indicates that the 'delivery\_plan' has been planned to satisfy the 'item\_request' but is in fact planned for more 'quantity' than requested. This Problem will not occur if the 'item\_request' has an infinite 'quantity' range.



<i>Extensions</i>	<i>REQUEST_PLANNED_EXCESS Extension</i>
-------------------	---

To resolve this Problem, either increase *item\_request.quantity* (unlikely), or reduce the Quantity planned by *item\_request.delivery\_plan* to be within the requested Quantity\_Range.

This is in both the REQUEST and REQUEST\_PLAN Problem\_Set categories, as well as REQUEST\_PLANNED\_EXCESS.

The REQUEST\_PLANNED\_EXCESS model has fields that references these models :  
*Item\_Request*, *Item\_Promise*, *Item\_Acceptance*, *Delivery\_Request*,  
*Delivery\_Promise*, *Delivery\_Acceptance*, *Operation\_Plan*.

*item\_request* - - *a Item\_Request field of model REQUEST\_PLANNED\_EXCESS*  
The *Item\_Request* that has this REQUEST\_PLANNED\_EXCESS Problem. This  
*item\_request.delivery\_plan* is planned to deliver more quantity than this  
*item\_request.quantity* range.  
Properties:   Export-Only Field

*item\_promise* - - *a Item\_Promise field of model REQUEST\_PLANNED\_EXCESS*  
The *Item\_Promise* for the *item\_request*.  
Properties:   Export-Only Field

*item\_acceptance* - - *a Item\_Acceptance field of model REQUEST\_PLANNED\_EXCESS*  
The *Item\_Acceptance* for the *item\_request*.  
Properties:   Export-Only Field

*delivery\_request* - - *a Delivery\_Request field of model REQUEST\_PLANNED\_EXCESS*  
This is simply shorthand for *item\_request.owner*.  
Properties:   Export-Only Field

*delivery\_promise* - - *a Delivery\_Promise field of model REQUEST\_PLANNED\_EXCESS*  
This is simply shorthand for *item\_promise.owner*.  
Properties:   Export-Only Field

*delivery\_acceptance* - - *a Delivery\_Acceptance field of model REQUEST\_PLANNED\_EXCESS*  
This is simply shorthand for *item\_acceptance.owner*.  
Properties:   Export-Only Field

<i>Extensions</i>	<i>PROMISE_NOT_PLANNED Extension</i>
-------------------	--------------------------------------

*request* - - *a Request field of model REQUEST\_PLANNED\_EXCESS*  
This is simply shorthand for *item\_request.delivery\_request.owner*.  
Properties:   Export-Only Field

*promise* - - *a Promise field of model REQUEST\_PLANNED\_EXCESS*  
This is simply shorthand for *item\_promise.delivery\_promise.owner*.  
Properties:   Export-Only Field

*acceptance* - - *a Acceptance field of model REQUEST\_PLANNED\_EXCESS*  
This is simply shorthand for *item\_acceptance.delivery\_acceptance.owner*.  
Properties:   Export-Only Field

*delivery\_plan* - - *a Operation\_Plan field of model REQUEST\_PLANNED\_EXCESS*  
This is simply shorthand for *item\_request.delivery\_plan*. This *delivery\_plan* is  
planned to deliver more than *item\_request.quantity*.  
Properties:   Export-Only Field

**PROMISE\_NOT\_PLANNED** - - *a category extension of model Problem*

A PROMISE\_NOT\_PLANNED Problem indicates that the *delivery\_plan* has not been planned to satisfy the *item\_promise*: either there is no *delivery\_plan*, or the *delivery\_plan* is for the *item\_request* rather than the *item\_promise*. This Problem will not occur if the *delivery\_promise* has an infinite due Date or the *item\_promise* is for a zero quantity.

To resolve this Problem, either eliminate or zero out the *item\_promise* (unlikely), or *plan\_to\_satisfy* the *item\_promise* which will create and/or replan the *delivery\_plan* to meet the *item\_promise*, or if the *delivery\_plan* is for the *item\_request*, then *promise\_as\_planned*.

This is in the PROMISE Problem\_Set category, as well as PROMISE\_NOT\_PLANNED.

The PROMISE\_NOT\_PLANNED model has fields that references these models :  
*Item\_Request*, *Item\_Promise*, *Item\_Acceptance*, *Delivery\_Request*,  
*Delivery\_Promise*, *Delivery\_Acceptance*, *Operation\_Plan*.

*item\_request* - - *a Item\_Request field of model PROMISE\_NOT\_PLANNED*  
The *Item\_Request* that has the PROMISE\_NOT\_PLANNED problem.  
Properties:   Export-Only Field

Extensions	PROMISE_NOT_PLANNED Extension
	<p><i>item_promise</i> - - <i>a</i> <i>Item_Promise</i> field of model <b>PROMISE_NOT_PLANNED</b> The <i>Item_Promise</i> that has this <b>PROMISE_NOT_PLANNED</b> Problem. This <i>item_promise.delivery_plan</i> either is nonexistent or is planned for this <i>item_promise.item_request</i>. Properties: Export-Only Field</p> <p><i>item_acceptance</i> - - <i>a</i> <i>Item_Acceptance</i> field of model <b>PROMISE_NOT_PLANNED</b> The <i>Item_Promise</i> for the <i>item_request</i>. Properties: Export-Only Field</p> <p><i>delivery_request</i> - - <i>a</i> <i>Delivery_Request</i> field of model <b>PROMISE_NOT_PLANNED</b> This is simply shorthand for <i>item_request.owner</i>. Properties: Export-Only Field</p> <p><i>delivery_promise</i> - - <i>a</i> <i>Delivery_Promise</i> field of model <b>PROMISE_NOT_PLANNED</b> This is simply shorthand for <i>item_promise.owner</i>. Properties: Export-Only Field</p> <p><i>delivery_acceptance</i> - - <i>a</i> <i>Delivery_Acceptance</i> field of model <b>PROMISE_NOT_PLANNED</b> This is simply shorthand for <i>item_acceptance.owner</i>. Properties: Export-Only Field</p> <p><i>request</i> - - <i>a</i> <i>Request</i> field of model <b>PROMISE_NOT_PLANNED</b> This is simply shorthand for <i>item_request.delivery_request.owner</i>. Properties: Export-Only Field</p> <p><i>promise</i> - - <i>a</i> <i>Promise</i> field of model <b>PROMISE_NOT_PLANNED</b> This is simply shorthand for <i>item_promise.delivery_promise.owner</i>. Properties: Export-Only Field</p> <p><i>acceptance</i> - - <i>a</i> <i>Acceptance</i> field of model <b>PROMISE_NOT_PLANNED</b> This is simply shorthand for <i>item_acceptance.delivery_acceptance.owner</i>. Properties: Export-Only Field</p> <p><i>delivery_plan</i> - - <i>a</i> <i>Operation_Plan</i> field of model <b>PROMISE_NOT_PLANNED</b> This is simply shorthand for <i>item_promise.delivery_plan</i>. Properties: Export-Only Field</p>

Extensions	PROMISE_PLANNED_LATE Extension
	<p><b>PROMISE_PLANNED_LATE</b> - - <i>a</i> category extension of model <i>Problem</i>  A <b>PROMISE_PLANNED_LATE</b> Problem indicates that the <i>'delivery_plan'</i> has been planned to satisfy the <i>'item_promise'</i> but is in fact planned later than the <i>'delivery_promise's due'</i> Dates. This Problem will not occur if the <i>'delivery_promise'</i> has an infinite <i>'due'</i> Date.  To resolve this Problem, either set the <i>'delivery_promise.due'</i> to be later (unlikely), or adjust the <i>'delivery_plan'</i> such that its end Date is within <i>'due'</i>.  This is in both the <b>PROMISE</b> and <b>PROMISE_PLAN</b> Problem_Set categories, as well as <b>PROMISE_PLANNED_LATE</b>.  The <b>PROMISE_PLANNED_LATE</b> model has fields that references these models : <i>Item_Request</i>, <i>Item_Promise</i>, <i>Item_Acceptance</i>, <i>Delivery_Request</i>, <i>Delivery_Promise</i>, <i>Delivery_Acceptance</i>, <i>Operation_Plan</i>.  <i>item_request</i> - - <i>a</i> <i>Item_Request</i> field of model <b>PROMISE_PLANNED_LATE</b> The <i>Item_Request</i> that has the <b>PROMISE_PLANNED_LATE</b> problem. Properties: Export-Only Field</p> <p><i>item_promise</i> - - <i>a</i> <i>Item_Promise</i> field of model <b>PROMISE_PLANNED_LATE</b> The <i>Item_Promise</i> that has this <b>PROMISE_PLANNED_LATE</b> Problem. This <i>item_promise.delivery_plan</i> is planned to end later than this <i>item_promise.owner.due</i> (the due Date of this <i>Item_Promise</i>). Properties: Export-Only Field</p> <p><i>item_acceptance</i> - - <i>a</i> <i>Item_Acceptance</i> field of model <b>PROMISE_PLANNED_LATE</b> The <i>Item_Acceptance</i> for the <i>item_request</i>. Properties: Export-Only Field</p> <p><i>delivery_request</i> - - <i>a</i> <i>Delivery_Request</i> field of model <b>PROMISE_PLANNED_LATE</b> This is simply shorthand for <i>item_request.owner</i>. Properties: Export-Only Field</p> <p><i>delivery_promise</i> - - <i>a</i> <i>Delivery_Promise</i> field of model <b>PROMISE_PLANNED_LATE</b> This is simply shorthand for <i>item_promise.owner</i>. The <i>'delivery_plan'</i> is planned to end later than this <i>'delivery_promise.due'</i>.</p>

<i>Extensions</i>	<i>PROMISE_PLANNED_EARLY Extension</i>
-------------------	--

Properties:      Export-Only Field

**delivery\_acceptance** - - *a Delivery\_Acceptance field of model*  
**PROMISE\_PLANNED\_LATE**

This is simply shorthand for 'item\_acceptance.owner'.

Properties:      Export-Only Field

**request** - - *a Request field of model* **PROMISE\_PLANNED\_LATE**

This is simply shorthand for 'item\_request.delivery\_request.owner'.

Properties:      Export-Only Field

**promise** - - *a Promise field of model* **PROMISE\_PLANNED\_LATE**

This is simply shorthand for 'item\_promise.delivery\_promise.owner'.

Properties:      Export-Only Field

**acceptance** - - *a Acceptance field of model* **PROMISE\_PLANNED\_LATE**

This is simply shorthand for 'item\_acceptance.delivery\_acceptance.owner'.

Properties:      Export-Only Field

**delivery\_plan** - - *a Operation\_Plan field of model* **PROMISE\_PLANNED\_LATE**

This is simply shorthand for 'item\_promise.delivery\_plan'. This 'delivery\_plan' is planned to end later than 'delivery\_promise.due'.

Properties:      Export-Only Field

**PROMISE\_PLANNED\_EARLY** - - *a category extension of model* Problem

A **PROMISE\_PLANNED\_EARLY** Problem indicates that the 'delivery\_plan' has been planned to satisfy the 'item\_promise' but is in fact planned earlier than the 'delivery\_promise's 'due' Dates. This Problem will not occur if the 'delivery\_promise' has an infinite 'due' Date.

To resolve this Problem, either set the 'delivery\_promise.due' to be earlier (unlikely), or adjust the 'delivery\_plan' such that its end Date is within 'due'.

This is in both the **PROMISE** and **PROMISE\_PLAN** Problem\_Set categories, as well as **PROMISE\_PLANNED\_EARLY**.

The **PROMISE\_PLANNED\_EARLY** model has fields that references these models :  
**Item\_Request**, **Item\_Promise**, **Item\_Acceptance**, **Delivery\_Request**,  
**Delivery\_Promise**, **Delivery\_Acceptance**, **Operation\_Plan**.

<i>Extensions</i>	<i>PROMISE_PLANNED_EARLY Extension</i>
-------------------	--

**item\_request** - - *a Item\_Request field of model* **PROMISE\_PLANNED\_EARLY**  
The **Item\_Request** that has this **PROMISE\_PLANNED\_EARLY** Problem.

Properties:      Export-Only Field

**item\_promise** - - *a Item\_Promise field of model* **PROMISE\_PLANNED\_EARLY**

The **Item\_Promise** that has this **PROMISE\_PLANNED\_EARLY** Problem. This 'item\_promise.delivery\_plan' is planned to end earlier than this 'item\_promise.owner.due' (the due Date of this **Item\_Promise**).

Properties:      Export-Only Field

**item\_acceptance** - - *a Item\_Acceptance field of model*  
**PROMISE\_PLANNED\_EARLY**

The **Item\_Acceptance** for the 'item\_request'.

Properties:      Export-Only Field

**delivery\_request** - - *a Delivery\_Request field of model*  
**PROMISE\_PLANNED\_EARLY**

This is simply shorthand for 'item\_request.owner'.

Properties:      Export-Only Field

**delivery\_promise** - - *a Delivery\_Promise field of model*  
**PROMISE\_PLANNED\_EARLY**

This is simply shorthand for 'item\_promise.owner'. The 'delivery\_plan' is planned to end earlier than this 'delivery\_promise.due'.

Properties:      Export-Only Field

**delivery\_acceptance** - - *a Delivery\_Acceptance field of model*  
**PROMISE\_PLANNED\_EARLY**

This is simply shorthand for 'item\_acceptance.owner'.

Properties:      Export-Only Field

**request** - - *a Request field of model* **PROMISE\_PLANNED\_EARLY**

This is simply shorthand for 'item\_request.delivery\_request.owner'.

Properties:      Export-Only Field

**promise** - - *a Promise field of model* **PROMISE\_PLANNED\_EARLY**

This is simply shorthand for 'item\_promise.delivery\_promise.owner'.

Properties:      Export-Only Field

**acceptance** - - *a Acceptance field of model* **PROMISE\_PLANNED\_EARLY**

This is simply shorthand for 'item\_acceptance.delivery\_acceptance.owner'.

Properties: Export-Only Field

delivery\_plan -- a Operation\_Plan field of model  
PROMISE\_PLANNED\_EARLY

This is simply shorthand for 'item\_promise.delivery\_plan'. This 'delivery\_plan' is planned to end earlier than 'delivery\_promise.due'.  
Properties: Export-Only Field

PROMISE\_PLANNED\_SHORT -- a category extension of model Problem

A PROMISE\_PLANNED\_SHORT Problem indicates that the 'delivery\_plan' has been planned to satisfy the 'item\_promise' but is in fact planned for less 'quantity' than promised. This Problem will not occur if the 'item\_promise' has an infinite 'quantity' range.

To resolve this Problem, either reduce 'item\_promise.quantity' (unlikely), or increase the Quantity planned by 'item\_promise.delivery\_plan' to be within the promised Quantity\_Range.

This is in both the PROMISE and PROMISE\_PLAN Problem\_Set categories, as well as PROMISE\_PLANNED\_SHORT.

The PROMISE\_PLANNED\_SHORT model has fields that references these models :  
Item\_Request, Item\_Promise, Item\_Acceptance, Delivery\_Request,  
Delivery\_Promise, Delivery\_Acceptance, Operation\_Plan.

item\_request -- a Item\_Request field of model PROMISE\_PLANNED\_SHORT  
The Item\_Request that has this PROMISE\_PLANNED\_SHORT Problem.  
Properties: Export-Only Field

item\_promise -- a Item\_Promise field of model PROMISE\_PLANNED\_SHORT  
The Item\_Promise that has this PROMISE\_PLANNED\_SHORT Problem. This  
'item\_promise.delivery\_plan' is planned to deliver less 'quantity' than this  
'item\_promise.quantity' range.  
Properties: Export-Only Field

item\_acceptance -- a Item\_Acceptance field of model  
PROMISE\_PLANNED\_SHORT  
The Item\_Acceptance for the 'item\_request'.  
Properties: Export-Only Field

delivery\_request -- a Delivery\_Request field of model  
PROMISE\_PLANNED\_SHORT

This is simply shorthand for 'item\_request.owner'.  
Properties: Export-Only Field

delivery\_promise -- a Delivery\_Promise field of model  
PROMISE\_PLANNED\_SHORT  
This is simply shorthand for 'item\_promise.owner'.  
Properties: Export-Only Field

delivery\_acceptance -- a Delivery\_Acceptance field of model  
PROMISE\_PLANNED\_SHORT  
This is simply shorthand for 'item\_acceptance.owner'.  
Properties: Export-Only Field

request -- a Request field of model PROMISE\_PLANNED\_SHORT  
This is simply shorthand for 'item\_request.delivery\_request.owner'.  
Properties: Export-Only Field

promise -- a Promise field of model PROMISE\_PLANNED\_SHORT  
This is simply shorthand for 'item\_promise.delivery\_promise.owner'.  
Properties: Export-Only Field

acceptance -- a Acceptance field of model PROMISE\_PLANNED\_SHORT  
This is simply shorthand for 'item\_acceptance.delivery\_acceptance.owner'.  
Properties: Export-Only Field

delivery\_plan -- a Operation\_Plan field of model  
PROMISE\_PLANNED\_SHORT  
This is simply shorthand for 'item\_promise.delivery\_plan'. This 'delivery\_plan' is  
planned to deliver less than 'item\_promise.quantity'.  
Properties: Export-Only Field

PROMISE\_PLANNED\_EXCESS -- a category extension of model Problem

A PROMISE\_PLANNED\_EXCESS Problem indicates that the 'delivery\_plan' has  
been planned to satisfy the 'item\_promise' but is in fact planned for more 'quantity'  
than promised. This Problem will not occur if the 'item\_promise' has an infinite 'quan-  
tity' range.

To resolve this Problem, either increase 'item\_promise.quantity' (unlikely), or reduce the Quantity planned by 'item\_promise.delivery\_plan' to be within the promised Quantity\_Range.

This is in both the PROMISE and PROMISE\_PLAN Problem\_Set categories, as well as PROMISE\_PLANNED\_EXCESS.

The PROMISE\_PLANNED\_EXCESS model has fields that references these models :  
Item\_Request, Item\_Promise, Item\_Acceptance, Delivery\_Request,  
Delivery\_Promise, Delivery\_Acceptance, Operation\_Plan.

item\_request - - a Item\_Request field of model PROMISE\_PLANNED\_EXCESS  
The Item\_Request that has this PROMISE\_PLANNED\_EXCESS Problem.

Properties: Export-Only Field

item\_promise - - a Item\_Promise field of model PROMISE\_PLANNED\_EXCESS  
The Item\_Promise that has this PROMISE\_PLANNED\_EXCESS Problem. This  
'item\_promise.delivery\_plan' is planned to deliver more 'quantity' than this  
'item\_promise.quantity' range.

Properties: Export-Only Field

item\_acceptance - - a Item\_Acceptance field of model  
PROMISE\_PLANNED\_EXCESS  
The Item\_Acceptance for the 'item\_request'.

Properties: Export-Only Field

delivery\_request - - a Delivery\_Request field of model  
PROMISE\_PLANNED\_EXCESS  
This is simply shorthand for 'item\_request.owner'.

Properties: Export-Only Field

delivery\_promise - - a Delivery\_Promise field of model  
PROMISE\_PLANNED\_EXCESS  
This is simply shorthand for 'item\_promise.owner'.

Properties: Export-Only Field

delivery\_acceptance - - a Delivery\_Acceptance field of model  
PROMISE\_PLANNED\_EXCESS  
This is simply shorthand for 'item\_acceptance.owner'.

Properties: Export-Only Field

request - - a Request field of model PROMISE\_PLANNED\_EXCESS  
This is simply shorthand for 'item\_request.delivery\_request.owner'.

Properties: Export-Only Field

promise - - a Promise field of model PROMISE\_PLANNED\_EXCESS  
This is simply shorthand for 'item\_promise.delivery\_promise.owner'.

Properties: Export-Only Field

acceptance - - a Acceptance field of model PROMISE\_PLANNED\_EXCESS  
This is simply shorthand for 'item\_acceptance.delivery\_acceptance.owner'.

Properties: Export-Only Field

delivery\_plan - - a Operation\_Plan field of model  
PROMISE\_PLANNED\_EXCESS

This is simply shorthand for 'item\_promise.delivery\_plan'. This 'delivery\_plan' is  
planned to deliver more than 'item\_promise.quantity'.

Properties: Export-Only Field

ACCEPTANCE\_NOT\_PLANNED - - a category extension of model Problem

A ACCEPTANCE\_NOT\_PLANNED Problem indicates that the 'item\_acceptance'  
was 'accepted' after the 'item\_promise' was offered - - and - - the 'delivery\_plan' has not  
been planned to satisfy the 'item\_acceptance'; either there is no 'delivery\_plan', or the  
'delivery\_plan' is for the older 'item\_promise'. This Problem will not occur if the  
'delivery\_acceptance' has an infinite 'due Date', the 'item\_acceptance' is for a zero  
'quantity', or the 'item\_promise' was offered after the 'item\_acceptance' was issued.

To resolve this Problem, either eliminate or zero out the 'item\_acceptance' (unlikely),  
'offer' a Promise, or 'plan\_to\_satisfy' the 'item\_acceptance' which will create and/or  
replan the 'delivery\_plan' to meet the 'item\_acceptance'.

This is in the ACCEPTANCE Problem\_Set category, as well as  
ACCEPTANCE\_NOT\_PLANNED.

The ACCEPTANCE\_NOT\_PLANNED model has fields that references these models  
Item\_Request, Item\_Promise, Item\_Acceptance, Delivery\_Request,  
Delivery\_Promise, Delivery\_Acceptance, Operation\_Plan.

Extensions

ACCEPTANCE\_NOT\_PLANNED Extension

item\_request - - a Item\_Request field of model ACCEPTANCE\_NOT\_PLANNED  
The Item\_Acceptance that has this ACCEPTANCE\_NOT\_PLANNED Problem. This  
item\_acceptance.delivery\_plan' either is nonexistent or is planned for this  
item\_acceptance.item\_promise', which is older than this Item\_Acceptance.  
Properties: Export-Only Field

item\_promise - - a Item\_Promise field of model  
ACCEPTANCE\_NOT\_PLANNED  
The Item\_Promise for the item\_request'.  
Properties: Export-Only Field

item\_acceptance - - a Item\_Acceptance field of model  
ACCEPTANCE\_NOT\_PLANNED  
The Item\_Acceptance for the item\_request'.  
Properties: Export-Only Field

delivery\_request - - a Delivery\_Request field of model  
ACCEPTANCE\_NOT\_PLANNED  
This is simply shorthand for item\_request.owner'.  
Properties: Export-Only Field

delivery\_promise - - a Delivery\_Promise field of model  
ACCEPTANCE\_NOT\_PLANNED  
This is simply shorthand for item\_promise.owner'.  
Properties: Export-Only Field

delivery\_acceptance - - a Delivery\_Acceptance field of model  
ACCEPTANCE\_NOT\_PLANNED  
This is simply shorthand for item\_acceptance.owner'.  
Properties: Export-Only Field

request - - a Request field of model ACCEPTANCE\_NOT\_PLANNED  
This is simply shorthand for item\_request.delivery\_request.owner'.  
Properties: Export-Only Field

promise - - a Promise field of model ACCEPTANCE\_NOT\_PLANNED  
This is simply shorthand for item\_promise.delivery\_promise.owner'.  
Properties: Export-Only Field

acceptance - - a Acceptance field of model ACCEPTANCE\_NOT\_PLANNED  
This is simply shorthand for item\_acceptance.delivery\_acceptance.owner'.

Extensions

ACCEPTANCE\_PLANNED\_LATE Extension

Properties: Export-Only Field  
delivery\_plan - - a Operation\_Plan field of model  
ACCEPTANCE\_NOT\_PLANNED  
This is simply shorthand for item\_acceptance.delivery\_plan'.  
Properties: Export-Only Field

ACCEPTANCE\_PLANNED\_LATE -- a category extension of model Problem

A ACCEPTANCE\_PLANNED\_LATE Problem indicates that the 'delivery\_plan' has  
been planned to satisfy the 'item\_acceptance' but is in fact planned later than the  
'delivery\_acceptance's 'due' Dates. This Problem will not occur if the  
'delivery\_acceptance' has an infinite 'due' Date.

To resolve this Problem, either set the 'delivery\_acceptance.due' to be later (unlikely),  
or adjust the 'delivery\_plan' such that its end Date is within 'due'.

This is in both the ACCEPTANCE and ACCEPTANCE\_PLAN Problem\_Set categories,  
as well as ACCEPTANCE\_PLANNED\_LATE.

The ACCEPTANCE\_PLANNED\_LATE model has fields that references these models

Item\_Request, Item\_Promise, Item\_Acceptance, Delivery\_Request,  
Delivery\_Promise, Delivery\_Acceptance, Operation\_Plan.

item\_request - - a Item\_Request field of model  
ACCEPTANCE\_PLANNED\_LATE  
The Item\_Acceptance that has this ACCEPTANCE\_PLANNED\_LATE Problem. This  
'item\_acceptance.delivery\_plan' is planned to end later than this  
'item\_acceptance.owner.due' (the due Date of this Item\_Acceptance).  
Properties: Export-Only Field

item\_promise - - a Item\_Promise field of model  
ACCEPTANCE\_PLANNED\_LATE  
The Item\_Promise for the item\_request'.  
Properties: Export-Only Field

item\_acceptance - - a Item\_Acceptance field of model  
ACCEPTANCE\_PLANNED\_LATE  
The Item\_Acceptance for the item\_request'.  
Properties: Export-Only Field

Extensions	ACCEPTANCE_PLANNED_EARLY Extension
------------	------------------------------------

delivery\_request -- a Delivery\_Request field of model  
ACCEPTANCE\_PLANNED\_LATE  
This is simply shorthand for item\_request.owner. The 'delivery\_plan' is planned to end later than this 'delivery\_acceptance.due'.  
Properties: Export-Only Field

delivery\_promise -- a Delivery\_Promise field of model  
ACCEPTANCE\_PLANNED\_LATE  
This is simply shorthand for item\_promise.owner.  
Properties: Export-Only Field

delivery\_acceptance -- a Delivery\_Acceptance field of model  
ACCEPTANCE\_PLANNED\_LATE  
This is simply shorthand for item\_acceptance.owner.  
Properties: Export-Only Field

request -- a Request field of model ACCEPTANCE\_PLANNED\_LATE  
This is simply shorthand for item\_request.delivery\_request.owner.  
Properties: Export-Only Field

promise -- a Promise field of model ACCEPTANCE\_PLANNED\_LATE  
This is simply shorthand for item\_promise.delivery\_promise.owner.  
Properties: Export-Only Field

acceptance -- a Acceptance field of model ACCEPTANCE\_PLANNED\_LATE  
This is simply shorthand for item\_acceptance.delivery\_acceptance.owner.  
Properties: Export-Only Field

delivery\_plan -- a Operation\_Plan field of model  
ACCEPTANCE\_PLANNED\_LATE  
This is simply shorthand for item\_acceptance.delivery\_plan. This 'delivery\_plan' is planned to end later than 'delivery\_acceptance.due'.  
Properties: Export-Only Field

ACCEPTANCE\_PLANNED\_EARLY -- a category extension of model Problem  
A ACCEPTANCE\_PLANNED\_EARLY Problem indicates that the 'delivery\_plan' has been planned to satisfy the 'delivery\_acceptance' but is in fact planned earlier than the 'delivery\_acceptance's 'due' Date. This Problem will not occur if the 'delivery\_acceptance' has an infinite 'due' Date.

Extensions	ACCEPTANCE_PLANNED_EARLY Extension
------------	------------------------------------

To resolve this Problem, either set the 'delivery\_acceptance.due' to be earlier (unlikely), or adjust the 'delivery\_plan' such that its end Date is within 'due'.

This is in both the ACCEPTANCE and ACCEPTANCE\_PLAN Problem\_Set categories, as well as ACCEPTANCE\_PLANNED\_EARLY.

The ACCEPTANCE\_PLANNED\_EARLY model has fields that references these models:

Item\_Request, Item\_Promise, Item\_Acceptance, Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance, Operation\_Plan.

item\_request -- a Item\_Request field of model  
ACCEPTANCE\_PLANNED\_EARLY  
The Item\_Acceptance that has this ACCEPTANCE\_PLANNED\_EARLY Problem. This item\_acceptance.delivery\_plan' is planned to end earlier than this item\_acceptance.owner.due' (the due Date of this Item\_Acceptance).  
Properties: Export-Only Field

item\_promise -- a Item\_Promise field of model  
ACCEPTANCE\_PLANNED\_EARLY  
The Item\_Promise for the 'item\_request'.  
Properties: Export-Only Field

item\_acceptance -- a Item\_Acceptance field of model  
ACCEPTANCE\_PLANNED\_EARLY  
The Item\_Acceptance for the 'item\_request'.  
Properties: Export-Only Field

delivery\_request -- a Delivery\_Request field of model  
ACCEPTANCE\_PLANNED\_EARLY  
This is simply shorthand for item\_request.owner. The 'delivery\_plan' is planned to end earlier than this 'delivery\_acceptance.due'.  
Properties: Export-Only Field

delivery\_promise -- a Delivery\_Promise field of model  
ACCEPTANCE\_PLANNED\_EARLY  
This is simply shorthand for item\_promise.owner.  
Properties: Export-Only Field

delivery\_acceptance -- a Delivery\_Acceptance field of model  
ACCEPTANCE\_PLANNED\_EARLY  
This is simply shorthand for item\_acceptance.owner.  
Properties: Export-Only Field

request -- a Request field of model ACCEPTANCE\_PLANNED\_EARLY  
This is simply shorthand for item\_request.delivery\_request.owner.  
Properties: Export-Only Field

promise -- a Promise field of model ACCEPTANCE\_PLANNED\_EARLY  
This is simply shorthand for item\_promise.delivery\_promise.owner.  
Properties: Export-Only Field

acceptance -- a Acceptance field of model ACCEPTANCE\_PLANNED\_EARLY  
This is simply shorthand for item\_acceptance.delivery\_acceptance.owner.  
Properties: Export-Only Field

delivery\_plan -- a Operation\_Plan field of model  
ACCEPTANCE\_PLANNED\_EARLY  
This is simply shorthand for item\_acceptance.delivery\_plan. This delivery\_plan is  
planned to end earlier than delivery\_acceptance.due.  
Properties: Export-Only Field

ACCEPTANCE\_PLANNED\_SHORT -- a category extension of model Problem

A ACCEPTANCE\_PLANNED\_SHORT Problem indicates that the delivery\_plan has  
been planned to satisfy the item\_acceptance but is in fact planned for less quantity  
than accepted. This Problem will not occur if the item\_acceptance has an infinite  
quantity range.

To resolve this Problem, either reduce item\_acceptance.quantity (unlikely), or  
increase the Quantity planned by item\_acceptance.delivery\_plan to be within the  
accepted Quantity\_Range.

This is in both the ACCEPTANCE and ACCEPTANCE\_PLAN Problem\_Set categories,  
as well as ACCEPTANCE\_PLANNED\_SHORT.

The ACCEPTANCE\_PLANNED\_SHORT model has fields that references these models:  
Item\_Request, Item\_Promise, Item\_Acceptance, Delivery\_Request,  
Delivery\_Promise, Delivery\_Acceptance, Operation\_Plan.

item\_request -- a Item\_Request field of model  
ACCEPTANCE\_PLANNED\_SHORT  
The item\_Acceptance that has this ACCEPTANCE\_PLANNED\_SHORT Problem.  
This item\_acceptance.delivery\_plan is planned to deliver less quantity than this  
item\_acceptance.quantity range.  
Properties: Export-Only Field

item\_promise -- a Item\_Promise field of model  
ACCEPTANCE\_PLANNED\_SHORT  
The item\_Promise for the item\_request.  
Properties: Export-Only Field

item\_acceptance -- a Item\_Acceptance field of model  
ACCEPTANCE\_PLANNED\_SHORT  
The item\_Acceptance for the item\_request.  
Properties: Export-Only Field

delivery\_request -- a Delivery\_Request field of model  
ACCEPTANCE\_PLANNED\_SHORT  
This is simply shorthand for item\_request.owner.  
Properties: Export-Only Field

delivery\_promise -- a Delivery\_Promise field of model  
ACCEPTANCE\_PLANNED\_SHORT  
This is simply shorthand for item\_promise.owner.  
Properties: Export-Only Field

delivery\_acceptance -- a Delivery\_Acceptance field of model  
ACCEPTANCE\_PLANNED\_SHORT  
This is simply shorthand for item\_acceptance.owner.  
Properties: Export-Only Field

request -- a Request field of model ACCEPTANCE\_PLANNED\_SHORT  
This is simply shorthand for item\_request.delivery\_request.owner.  
Properties: Export-Only Field

promise -- a Promise field of model ACCEPTANCE\_PLANNED\_SHORT  
This is simply shorthand for item\_promise.delivery\_promise.owner.  
Properties: Export-Only Field



Extensions	ACCEPTANCE_PLANNED_EXCESS Extension
------------	-------------------------------------

acceptance -- *a Acceptance field of model ACCEPTANCE\_PLANNED\_SHORT*  
This is simply shorthand for 'item\_acceptance.delivery\_acceptance.owner'.  
Properties: Export-Only Field

delivery\_plan -- *a Operation\_Plan field of model ACCEPTANCE\_PLANNED\_SHORT*  
This is simply shorthand for 'item\_acceptance.delivery\_plan'. This 'delivery\_plan' is planned to deliver less than 'item\_acceptance.quantity'.  
Properties: Export-Only Field

**ACCEPTANCE\_PLANNED\_EXCESS -- a category extension of model Problem**

A ACCEPTANCE\_PLANNED\_EXCESS Problem indicates that the 'delivery\_plan' has been planned to satisfy the 'item\_acceptance' but is in fact planned for more 'quantity' than accepted. This Problem will not occur if the 'item\_acceptance' has an infinite 'quantity' range.

To resolve this Problem, either increase 'item\_acceptance.quantity' (unlikely), or reduce the Quantity planned by 'item\_acceptance.delivery\_plan' to be within the accepted Quantity\_Range.

This is in both the ACCEPTANCE and ACCEPTANCE\_PLAN Problem\_Set categories, as well as ACCEPTANCE\_PLANNED\_EXCESS.

The ACCEPTANCE\_PLANNED\_EXCESS model has fields that references these models :

Item\_Request, Item\_Promise, Item\_Acceptance, Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance, Operation\_Plan,

Item\_request -- *a Item\_Request field of model ACCEPTANCE\_PLANNED\_EXCESS*

The Item\_Acceptance that has this ACCEPTANCE\_PLANNED\_EXCESS Problem.

This 'item\_acceptance.delivery\_plan' is planned to deliver more 'quantity' than this 'item\_acceptance.quantity' range.

Properties: Export-Only Field

Item\_promise -- *a Item\_Promise field of model ACCEPTANCE\_PLANNED\_EXCESS*

The Item\_Promise for the 'item\_request'.

Properties: Export-Only Field

Extensions	PLANNED_BEFORE_CURRENT Extension
------------	----------------------------------

item\_acceptance -- *a Item\_Acceptance field of model ACCEPTANCE\_PLANNED\_EXCESS*  
The Item\_Acceptance for the 'item\_request'.  
Properties: Export-Only Field

delivery\_request -- *a Delivery\_Request field of model ACCEPTANCE\_PLANNED\_EXCESS*  
This is simply shorthand for 'item\_request.owner'.  
Properties: Export-Only Field

delivery\_promise -- *a Delivery\_Promise field of model ACCEPTANCE\_PLANNED\_EXCESS*  
This is simply shorthand for 'item\_promise.owner'.  
Properties: Export-Only Field

delivery\_acceptance -- *a Delivery\_Acceptance field of model ACCEPTANCE\_PLANNED\_EXCESS*  
This is simply shorthand for 'item\_acceptance.owner'.  
Properties: Export-Only Field

request -- *a Request field of model ACCEPTANCE\_PLANNED\_EXCESS*  
This is simply shorthand for 'item\_request.delivery\_request.owner'.  
Properties: Export-Only Field

promise -- *a Promise field of model ACCEPTANCE\_PLANNED\_EXCESS*  
This is simply shorthand for 'item\_promise.delivery\_promise.owner'.  
Properties: Export-Only Field

acceptance -- *a Acceptance field of model ACCEPTANCE\_PLANNED\_EXCESS*  
This is simply shorthand for 'item\_acceptance.delivery\_acceptance.owner'.  
Properties: Export-Only Field

delivery\_plan -- *a Operation\_Plan field of model ACCEPTANCE\_PLANNED\_EXCESS*

This is simply shorthand for 'item\_acceptance.delivery\_plan'. This 'delivery\_plan' is planned to deliver more than 'item\_acceptance.quantity'.

Properties: Export-Only Field

**PLANNED\_BEFORE\_CURRENT -- a category extension of model Problem**

<i>Extensions</i>	<i>UNRELEASED Extension</i>
-------------------	-----------------------------

A **PLANNED\_BEFORE\_CURRENT** Problem indicates that the 'operation\_plan' has been planned such that it starts before the plan.current date. Only the top operation plan of an operation plan tree may have this problem.

The **PLANNED\_BEFORE\_CURRENT** model has fields that references these models :

Operation\_Plan.

operation\_plan - - *a Operation\_Plan field of model PLANNED\_BEFORE\_CURRENT*  
The Operation\_Plan with this **PLANNED\_BEFORE\_CURRENT** Problem.  
Properties: Export-Only Field

**UNRELEASED** -- *a category extension of model Problem*

An **UNRELEASED** Problem indicates that the 'operation\_plan' has been planned such that it starts before the release\_fence and is not released. Only the top operation plan of an operation plan tree may have this problem.

The **UNRELEASED** model has fields that references these models :

Operation\_Plan.

operation\_plan - - *a Operation\_Plan field of model UNRELEASED*  
The Operation\_Plan with this **UNRELEASED** Problem.  
Properties: Export-Only Field

**NEEDS\_RELEASE** -- *a category extension of model Problem*

A **NEEDS\_RELEASE** Problem indicates that the 'operation\_plan' has been planned such that it starts before the release\_soon\_fence and has not been released. Only the top operation plan of an operation plan tree may have this problem.

The **NEEDS\_RELEASE** model has fields that references these models :

Operation\_Plan.

operation\_plan - - *a Operation\_Plan field of model NEEDS\_RELEASE*  
The Operation\_Plan with this **NEEDS\_RELEASE** Problem.  
Properties: Export-Only Field

**INCONSISTENT\_OPPLAN** -- *a category extension of model Problem*

<i>Extensions</i>	<i>REQUEST_PROMISED_LATE Extension</i>
-------------------	--

A **INCONSISTENT\_OPPLAN** Problem indicates that an 'operation\_plan' should have been replanned, but was not, because to do so would have required replanning an operation plan which had locked\_as\_planned set to 'true'. Note that the **INCONSISTENT\_OPPLAN** problem is associated with the operation plan that could not be replanned; it may or may not have its own locked\_as\_planned set to 'true'. This problem arises only when some property of an operation changes (per-unit time, for example) and the necessary replanning was prevented because some of the operation plans that needed to be replanned had locked\_as\_planned set to 'true'. An example of when this can happen would be in a routing with three operations. An operation plan for this routing may have the two "quiet" sub-operation plans with locked\_as\_planned set to 'true'. If the per-unit time of the middle sub-operation plan is changed and the necessary replanning would cause the outer sub-operation plans to be changed, **INCONSISTENT\_OPPLAN** problems would be created.

Note that the resolver for the **INCONSISTENT\_OPPLAN** problem does nothing; the only way to remove the **INCONSISTENT\_OPPLAN** problem is to set locked\_as\_planned to 'false' on the operation plan(s) which prevented the replanning. When a process extension is changed and there are associated operation plans with that operation with locked\_as\_planned set to 'true', undoing the change of process extension may leave extraneous **INCONSISTENT\_OPPLAN** problems. This is a known limitation in undoing process extension changes in the presence of associated operation\_plans with locked\_as\_planned set to 'true'. Unlocking those operation plans will eliminate the problems.

The **INCONSISTENT\_OPPLAN** model has fields that references these models :

Operation\_Plan.

operation\_plan - - *a Operation\_Plan field of model INCONSISTENT\_OPPLAN*  
The Operation\_Plan with this **INCONSISTENT\_OPPLAN** Problem.  
Properties: Export-Only Field

**REQUEST\_PROMISED\_LATE** -- *a category extension of model Problem*

A **REQUEST\_PROMISED\_LATE** Problem indicates that the 'item\_promise' has 'due' Dates that are later than the 'due' Dates of its 'item\_request'. The Promise is for later than requested. To resolve this Problem, either the Request must be made for later (unlikely), or the Promise must be offered for earlier.

This is in the **REQUEST\_PROMISE**, and **REQUEST\_PROMISE\_Problem\_Set** categories, as well as **REQUEST\_PROMISED\_LATE**.

The **REQUEST\_PROMISED\_LATE** model has fields that references these models :  
Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance.

Extensions	REQUEST_PROMISED_EARLY Extension
------------	----------------------------------

delivery\_request -- a Delivery\_Request field of model  
REQUEST\_PROMISED\_LATE  
The Delivery\_Request whose delivery\_promise has later 'due' Dates.  
Properties: Export-Only Field

delivery\_promise -- a Delivery\_Promise field of model  
REQUEST\_PROMISED\_LATE  
The Delivery\_Promise whose 'due' Dates are later than its 'delivery\_request'.  
Properties: Export-Only Field

delivery\_acceptance -- a Delivery\_Acceptance field of model  
REQUEST\_PROMISED\_LATE  
The Delivery\_Acceptance for the 'delivery\_request'.  
Properties: Export-Only Field

request -- a Request field of model REQUEST\_PROMISED\_LATE  
This is simply shorthand for 'delivery\_request.owner'.  
Properties: Export-Only Field

promise -- a Promise field of model REQUEST\_PROMISED\_LATE  
This is simply shorthand for 'delivery\_promise.owner'.  
Properties: Export-Only Field

acceptance -- a Acceptance field of model REQUEST\_PROMISED\_LATE  
This is simply shorthand for 'delivery\_acceptance.owner'.  
Properties: Export-Only Field

REQUEST\_PROMISED\_EARLY -- a category extension of model Problem

A REQUEST\_PROMISED\_EARLY Problem indicates that the 'delivery\_promise' has earlier 'due' Dates than its 'delivery\_request'. The Promise is for earlier than requested. To resolve this Problem, either the Request must be made for earlier or the Promise for later.

This is in the REQUEST\_PROMISE, and REQUEST\_PROMISE\_Problem\_Set categories, as well as REQUEST\_PROMISED\_EARLY.

The REQUEST\_PROMISED\_EARLY model has fields that references these models: Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance.

Extensions	REQUEST_PROMISED_SHORT Extension
------------	----------------------------------

delivery\_request -- a Delivery\_Request field of model  
REQUEST\_PROMISED\_EARLY  
The Delivery\_Request whose 'delivery\_promise' has earlier 'due' Dates.  
Properties: Export-Only Field

delivery\_promise -- a Delivery\_Promise field of model  
REQUEST\_PROMISED\_EARLY  
The Delivery\_Promise whose 'due' Dates are earlier than its 'delivery\_request'.  
Properties: Export-Only Field

delivery\_acceptance -- a Delivery\_Acceptance field of model  
REQUEST\_PROMISED\_EARLY  
The Delivery\_Acceptance for the 'delivery\_request'.  
Properties: Export-Only Field

request -- a Request field of model REQUEST\_PROMISED\_EARLY  
This is simply shorthand for 'delivery\_request.owner'.  
Properties: Export-Only Field

promise -- a Promise field of model REQUEST\_PROMISED\_EARLY  
This is simply shorthand for 'delivery\_promise.owner'.  
Properties: Export-Only Field

acceptance -- a Acceptance field of model REQUEST\_PROMISED\_EARLY  
This is simply shorthand for 'delivery\_acceptance.owner'.  
Properties: Export-Only Field

REQUEST\_PROMISED\_SHORT -- a category extension of model Problem

A REQUEST\_PROMISED\_SHORT Problem indicates that the 'item\_promise' has less 'quantity' than its 'item\_request'. The Promise is for less than requested. To resolve this Problem, either the Request must be made for less or the Promise for more.

This is in the REQUEST\_PROMISE, and REQUEST\_PROMISE\_Problem\_Set categories, as well as REQUEST\_PROMISED\_SHORT.

The REQUEST\_PROMISED\_SHORT model has fields that references these models: Item\_Request, Item\_Promise, Item\_Acceptance, Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance.

**item\_request** - - *a Item\_Request field of model REQUEST\_PROMISED\_SHORT*  
The Item\_Request whose item\_promise has less 'quantity'.  
Properties: Export-Only Field

**item\_promise** - - *a Item\_Promise field of model REQUEST\_PROMISED\_SHORT*  
The Item\_Promise whose 'quantity' is less than its item\_request.  
Properties: Export-Only Field

**item\_acceptance** - - *a Item\_Acceptance field of model REQUEST\_PROMISED\_SHORT*  
The Item\_Acceptance for the item\_request.  
Properties: Export-Only Field

**delivery\_request** - - *a Delivery\_Request field of model REQUEST\_PROMISED\_SHORT*  
This is simply shorthand for item\_request.owner.  
Properties: Export-Only Field

**delivery\_promise** - - *a Delivery\_Promise field of model REQUEST\_PROMISED\_SHORT*  
This is simply shorthand for item\_promise.owner.  
Properties: Export-Only Field

**delivery\_acceptance** - - *a Delivery\_Acceptance field of model REQUEST\_PROMISED\_SHORT*  
This is simply shorthand for item\_acceptance.owner.  
Properties: Export-Only Field

**request** - - *a Request field of model REQUEST\_PROMISED\_SHORT*  
This is simply shorthand for item\_request.delivery\_request.owner.  
Properties: Export-Only Field

**promise** - - *a Promise field of model REQUEST\_PROMISED\_SHORT*  
This is simply shorthand for item\_promise.delivery\_promise.owner.  
Properties: Export-Only Field

**acceptance** - - *a Acceptance field of model REQUEST\_PROMISED\_SHORT*  
This is simply shorthand for item\_acceptance.delivery\_acceptance.owner.  
Properties: Export-Only Field

#### REQUEST\_PROMISED\_EXCESS - - *a category extension of model Problem*

A REQUEST\_PROMISED\_EXCESS Problem indicates that the item\_promise has more 'quantity' than its item\_request. The Promise is for more than requested. To resolve this Problem, either the Request must be made for more or the Promise for made for less.

This is in the REQUEST\_PROMISE, and REQUEST\_PROMISE\_Problem\_Set categories, as well as REQUEST\_PROMISED\_EXCESS.

The REQUEST\_PROMISED\_EXCESS model has fields that references these models:  
Item\_Request, Item\_Promise, Item\_Acceptance, Delivery\_Request,  
Delivery\_Promise, Delivery\_Acceptance.

**item\_request** - - *a Item\_Request field of model REQUEST\_PROMISED\_EXCESS*  
The Item\_Request whose item\_promise has excess 'quantity'.  
Properties: Export-Only Field

**item\_promise** - - *a Item\_Promise field of model REQUEST\_PROMISED\_EXCESS*  
The Item\_Promise whose 'quantity' is more than its item\_request.  
Properties: Export-Only Field

**item\_acceptance** - - *a Item\_Acceptance field of model REQUEST\_PROMISED\_EXCESS*  
The Item\_Acceptance for the item\_request.  
Properties: Export-Only Field

**delivery\_request** - - *a Delivery\_Request field of model REQUEST\_PROMISED\_EXCESS*  
This is simply shorthand for item\_request.owner.  
Properties: Export-Only Field

**delivery\_promise** - - *a Delivery\_Promise field of model REQUEST\_PROMISED\_EXCESS*  
This is simply shorthand for item\_promise.owner.  
Properties: Export-Only Field

Extensions	ITEM_PROMISE_OVERPRICED Extension
------------	-----------------------------------

delivery\_acceptance -- a Delivery Acceptance field of model  
REQUEST\_PROMISED\_EXCESS

This is simply shorthand for item\_acceptance.owner.  
Properties: Export-Only Field

request -- a Request field of model REQUEST\_PROMISED\_EXCESS  
This is simply shorthand for item\_request.delivery\_request.owner.  
Properties: Export-Only Field

promise -- a Promise field of model REQUEST\_PROMISED\_EXCESS  
This is simply shorthand for item\_promise.delivery\_promise.owner.  
Properties: Export-Only Field

acceptance -- a Acceptance field of model REQUEST\_PROMISED\_EXCESS  
This is simply shorthand for item\_acceptance.delivery\_acceptance.owner.  
Properties: Export-Only Field

ITEM\_PROMISE\_OVERPRICED -- a category extension of model Problem

An ITEM\_PROMISE\_OVERPRICED Problem indicates that the item\_promise has a higher price than its item\_request's max\_price. The promise is more expensive than requested. To resolve this Problem, either the Request must be made for more or the Promise for made for less. The latter is often done by setting discount.

This is in the REQUEST\_PROMISE and REQUEST\_PROMISE Problem Set categories, as well as ITEM\_PROMISE\_OVERPRICED.

The ITEM\_PROMISE\_OVERPRICED model has fields that references these models:  
Item\_Request, Item\_Promise, Item\_Acceptance, Delivery\_Request,  
Delivery\_Promise, Delivery\_Acceptance.

item\_request -- a Item\_Request field of model  
ITEM\_PROMISE\_OVERPRICED

The item\_Request whose item\_promise has excess quantity.  
Properties: Export-Only Field

item\_promise -- a Item\_Promise field of model  
ITEM\_PROMISE\_OVERPRICED

The item\_Promise whose quantity is more than its item\_request.  
Properties: Export-Only Field

Extensions	DELIVERY_PROMISE_OVERPRICED Extension
------------	---------------------------------------

item\_acceptance -- a Item Acceptance field of model  
ITEM\_PROMISE\_OVERPRICED

Item Acceptance for the item\_request.  
Properties: Export-Only Field

delivery\_request -- a Delivery\_Request field of model  
ITEM\_PROMISE\_OVERPRICED  
This is simply shorthand for item\_request.owner.  
Properties: Export-Only Field

delivery\_promise -- a Delivery\_Promise field of model  
ITEM\_PROMISE\_OVERPRICED  
This is simply shorthand for item\_promise.owner.  
Properties: Export-Only Field

delivery\_acceptance -- a Delivery\_Acceptance field of model  
ITEM\_PROMISE\_OVERPRICED  
This is simply shorthand for item\_acceptance.owner.  
Properties: Export-Only Field

request -- a Request field of model ITEM\_PROMISE\_OVERPRICED  
This is simply shorthand for item\_request.delivery\_request.owner.  
Properties: Export-Only Field

promise -- a Promise field of model ITEM\_PROMISE\_OVERPRICED  
This is simply shorthand for item\_promise.delivery\_promise.owner.  
Properties: Export-Only Field

acceptance -- a Acceptance field of model ITEM\_PROMISE\_OVERPRICED  
This is simply shorthand for item\_acceptance.delivery\_acceptance.owner.  
Properties: Export-Only Field

DELIVERY\_PROMISE\_OVERPRICED -- a category extension of model Problem

A DELIVERY\_PROMISE\_OVERPRICED Problem indicates that the delivery\_promise has a higher delivery\_price than its delivery\_request's max\_price. The promise is more expensive than requested. To resolve this Problem, either the Request must be made for more or the Promise for made for less. The latter is often done by setting delivery\_discount.

This is in the REQUEST, PROMISE, and REQUEST\_PROMISE Problem\_Set categories, as well as DELIVERY\_PROMISE\_OVERPRICED.

The DELIVERY\_PROMISE\_OVERPRICED model has fields that references these models :

Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance.

delivery\_request -- a Delivery\_Request field of model DELIVERY\_PROMISE\_OVERPRICED

The Delivery\_Request whose 'delivery\_promise' has excess price:

Properties: Export-Only Field

delivery\_promise -- a Delivery\_Promise field of model DELIVERY\_PROMISE\_OVERPRICED

The Delivery\_Promise whose 'price' is more than its 'delivery\_request.max\_price':

Properties: Export-Only Field

delivery\_acceptance -- a Delivery\_Acceptance field of model DELIVERY\_PROMISE\_OVERPRICED

The Delivery\_Acceptance for the 'delivery\_request'

Properties: Export-Only Field

request -- a Request field of model DELIVERY\_PROMISE\_OVERPRICED

This is simply shorthand for 'delivery\_request.owner':

Properties: Export-Only Field

promise -- a Promise field of model DELIVERY\_PROMISE\_OVERPRICED

This is simply shorthand for 'delivery\_promise.owner':

Properties: Export-Only Field

acceptance -- a Acceptance field of model DELIVERY\_PROMISE\_OVERPRICED

This is simply shorthand for 'delivery\_acceptance.owner':

Properties: Export-Only Field

A PROMISE\_NOT\_OFFERED Problem indicates that a Request has been issued, but the Promise has not been offered. More precisely, 'promise.date\_offered' is before 'request.date\_issued'. To resolve, either the Promise field 'date\_offered' must be set to 'date\_issued' or later.

This is in the REQUEST and PROMISE Problem\_Set categories, as well as PROMISE\_NOT\_OFFERED.

The PROMISE\_NOT\_OFFERED model has fields that references these models :  
Request, Promise, Acceptance.

request -- a Request field of model PROMISE\_NOT\_OFFERED

The Request with this PROMISE\_NOT\_OFFERED Problem.

Properties: Export-Only Field

promise -- a Promise field of model PROMISE\_NOT\_OFFERED

The Promise for the 'request':

Properties: Export-Only Field

acceptance -- a Acceptance field of model PROMISE\_NOT\_OFFERED

The Acceptance for the 'request':

Properties: Export-Only Field

A PROMISE\_NOT\_ACCEPTED Problem indicates that a Promise has been offered but not yet been accepted. More precisely, 'date\_accepted' is before 'date\_offered', which is at or after 'date\_issued'. To resolve, the field 'date\_accepted' must be set to 'date\_offered' or later.

This is in the REQUEST and PROMISE Problem\_Set categories, as well as PROMISE\_NOT\_ACCEPTED.

The PROMISE\_NOT\_ACCEPTED model has fields that references these models :  
Request, Promise, Acceptance.

request -- a Request field of model PROMISE\_NOT\_ACCEPTED

The Request that has this PROMISE\_NOT\_ACCEPTED Problem.

Properties: Export-Only Field

promise -- a Promise field of model PROMISE\_NOT\_ACCEPTED

The Promise with this PROMISE\_NOT\_ACCEPTED Problem.

Properties: Export-Only Field

Extensions	ACCEPTANCE_INCONSISTENT Extension
------------	-----------------------------------

acceptance - - *a* Acceptance field of model PROMISE\_NOT\_ACCEPTED  
The Acceptance for the 'request'.  
Properties: Export-Only Field

ACCEPTANCE\_INCONSISTENT -- *a* category extension of model Problem

A ACCEPTANCE\_INCONSISTENT Problem indicates that a Promise has been 'accepted' but the acceptance date or quantity is different from what had been promised  
This is in the REQUEST and PROMISE Problem\_Set categories, as well as ACCEPTANCE\_INCONSISTENT.

The ACCEPTANCE\_INCONSISTENT model has fields that references these models :  
Request, Promise, Acceptance.

request - - *a* Request field of model ACCEPTANCE\_INCONSISTENT  
The Request that has this ACCEPTANCE\_INCONSISTENT Problem.  
Properties: Export-Only Field

promise - - *a* Promise field of model ACCEPTANCE\_INCONSISTENT  
The Promise for the 'request'.  
Properties: Export-Only Field

acceptance - - *a* Acceptance field of model ACCEPTANCE\_INCONSISTENT  
The Acceptance with this ACCEPTANCE\_INCONSISTENT Problem.  
Properties: Export-Only Field

REQUEST\_QUEUED -- *a* category extension of model Problem

A REQUEST\_QUEUED Problem indicates that a Request has been answered unsatisfactorily, and then 'queued' by the requestor for later reconsideration. More precisely, 'date\_queued' is at or after 'date\_offered', which is at or after 'date\_issued'. To resolve, 'date\_issued' or the Promise field 'date\_offered' must be set to at or after 'date\_queued'.

This is in the REQUEST Problem\_Set category, as well as REQUEST\_QUEUED.

The REQUEST\_QUEUED model has fields that references these models :  
Request, Promise, Acceptance.

request - - *a* Request field of model REQUEST\_QUEUED  
The Request with this REQUEST\_QUEUED Problem.

Extensions	DELIVERY_REQUEST_NOT_COORDINATED Extension
------------	--

Properties: Export-Only Field  
promise - - *a* Promise field of model REQUEST\_QUEUED  
The Promise for the 'request'.  
Properties: Export-Only Field

acceptance - - *a* Acceptance field of model REQUEST\_QUEUED  
The Acceptance for the 'request'.  
Properties: Export-Only Field

DELIVERY\_REQUEST\_NOT\_COORDINATED -- *a* category extension of model Problem

A DELIVERY\_REQUEST\_NOT\_COORDINATED Problem indicates that the items of a Delivery\_Request are not all scheduled for delivery on the same date.

This type of problem is resolved as follows. First, flip a coin to decide whether to correct earliness/lateness. If the coin toss indicates correcting earliness/lateness, then move all early line item deliveries out to the start of the delivery's due period, and move all late line item deliveries in to the end of that period. Whether correcting earliness/lateness or not, proceed by deciding whether to coordinate line item deliveries by either moving in or moving out. If moving in, then move all line item deliveries to the date of the earliest line item delivery. If moving out, then move all line item deliveries to the date of the latest delivery.

The DELIVERY\_REQUEST\_NOT\_COORDINATED model has fields that references these models :  
Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance.

delivery\_request - - *a* Delivery\_Request field of model DELIVERY\_REQUEST\_NOT\_COORDINATED  
The Delivery\_Request with this DELIVERY\_REQUEST\_NOT\_COORDINATED Problem.

Properties: Export-Only Field

delivery\_promise - - *a* Delivery\_Promise field of model DELIVERY\_REQUEST\_NOT\_COORDINATED  
The Delivery\_Promise for the 'delivery\_request'.  
Properties: Export-Only Field

delivery\_acceptance -- a Delivery\_Acceptance field of model  
DELIVERY\_REQUEST\_NOT\_COORDINATED  
The Delivery\_Acceptance for the 'delivery\_request'.  
Properties: Export-Only Field

request -- a Request field of model  
DELIVERY\_REQUEST\_NOT\_COORDINATED  
This is simply shorthand for 'delivery\_request.owner'.  
Properties: Export-Only Field

promise -- a Promise field of model  
DELIVERY\_REQUEST\_NOT\_COORDINATED  
This is simply shorthand for 'delivery\_promise.owner'.  
Properties: Export-Only Field

acceptance -- a Acceptance field of model  
DELIVERY\_REQUEST\_NOT\_COORDINATED  
This is simply shorthand for 'delivery\_acceptance.owner'.  
Properties: Export-Only Field

DELIVERY\_PROMISE\_NOT\_COORDINATED -- a category extension of model Problem

A DELIVERY\_PROMISE\_NOT\_COORDINATED Problem indicates that the items of a Delivery\_Promise are not all scheduled for delivery on the same date.

For an explanation of how this problem is resolved, see the DELIVERY\_REQUEST\_NOT\_COORDINATED problem.

The DELIVERY\_PROMISE\_NOT\_COORDINATED model has fields that references these models :  
Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance.

delivery\_request -- a Delivery\_Request field of model  
DELIVERY\_PROMISE\_NOT\_COORDINATED  
The Delivery\_Request that has this DELIVERY\_NOT\_COORDINATED Problem.  
Properties: Export-Only Field

delivery\_promise -- a Delivery\_Promise field of model  
DELIVERY\_PROMISE\_NOT\_COORDINATED  
The Delivery\_Promise with this DELIVERY\_PROMISE\_NOT\_COORDINATED Problem.

Properties: Export-Only Field

delivery\_acceptance -- a Delivery\_Acceptance field of model  
DELIVERY\_PROMISE\_NOT\_COORDINATED  
The Delivery\_Acceptance for the 'delivery\_request'.  
Properties: Export-Only Field

request -- a Request field of model  
DELIVERY\_PROMISE\_NOT\_COORDINATED  
This is simply shorthand for 'delivery\_request.owner'.  
Properties: Export-Only Field

promise -- a Promise field of model  
DELIVERY\_PROMISE\_NOT\_COORDINATED  
This is simply shorthand for 'delivery\_promise.owner'.  
Properties: Export-Only Field

acceptance -- a Acceptance field of model  
DELIVERY\_PROMISE\_NOT\_COORDINATED  
This is simply shorthand for 'delivery\_acceptance.owner'.  
Properties: Export-Only Field

DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED -- a category extension of model Problem

A DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED Problem indicates that the items of a Delivery\_Acceptance are not all scheduled for delivery on the same date.

For an explanation of how this problem is resolved, see the DELIVERY\_REQUEST\_NOT\_COORDINATED problem.

The DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED model has fields that references these models :  
Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance.

delivery\_request -- a Delivery\_Request field of model  
DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED  
The Delivery\_Request that has this  
DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED Problem.  
Properties: Export-Only Field



delivery\_promise - - *a Delivery\_Promise field of model*  
DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED  
The Delivery\_Promise for the 'delivery\_request'.  
Properties: Export-Only Field

delivery\_acceptance - - *a Delivery\_Acceptance field of model*  
DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED  
The Delivery\_Acceptance with this  
DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED Problem.  
Properties: Export-Only Field

request - - *a Request field of model*  
DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED  
This is simply shorthand for 'delivery\_request.owner'.  
Properties: Export-Only Field

promise - - *a Promise field of model*  
DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED  
This is simply shorthand for 'delivery\_promise.owner'.  
Properties: Export-Only Field

acceptance - - *a Acceptance field of model*  
DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED  
This is simply shorthand for 'delivery\_acceptance.owner'.  
Properties: Export-Only Field

NEGATIVE\_ATP -- *a category extension of model Problem*

A NEGATIVE\_ATP Problem indicates that the 'available\_to\_promise' quantity is negative for a particular Forecast\_Entry. This occurs when more Product is consumed from the Forecast\_Entry than was 'allocated'. This situation may arise when one of two things happens: (1) the 'allocated' quantity is reduced, or (2) the consumption increases. Note, however, that not all of this consumption need occur in the Forecast\_Entry which has a problem. Some of the 'available\_to\_promise' may have been consumed from a later Forecast\_Entry (if the Product for the Forecast\_Entry specifies a 'consume\_earlier' which allows this case.)

There is no automated resolution of this Problem. The allocation will need to be increased, or the actual promises that are consuming the allocation will need to be reduced.

The NEGATIVE\_ATP model has fields that references these models :

Forecast\_Entry.  
forecast\_entry - - *a Forecast\_Entry field of model NEGATIVE\_ATP*  
The Forecast\_Entry in which this problem occurred.  
Properties: Export-Only Field

NEGATIVE\_PLANNED\_ATP -- *a category extension of model Problem*

A NEGATIVE\_PLANNED\_ATP Problem indicates that the 'planned\_available' quantity is negative for a particular Forecast\_Entry. This occurs when more Product is consumed from the Forecast\_Entry than was 'planned'. This situation may arise when one of two things happens: (1) the 'planned' quantity is reduced, or (2) the consumption increases. Note, however, that not all of this consumption need occur in the Forecast\_Entry which has a problem. Some of the 'planned\_available' may have been consumed from a later Forecast\_Entry (if the Product for the Forecast\_Entry specifies a 'consume\_earlier' which allows this case.)

There is no automated resolution of this Problem. The allocation will need to be increased, or the actual promises that are consuming the allocation will need to be reduced.

The NEGATIVE\_PLANNED\_ATP model has fields that references these models :  
Forecast\_Entry.

forecast\_entry - - *a Forecast\_Entry field of model NEGATIVE\_PLANNED\_ATP*  
The Forecast\_Entry in which this problem occurred.

Properties: Export-Only Field

OVER\_COMMITTED -- *a category extension of model Problem*

An OVER\_COMMITTED Problem indicates that the 'committed' quantity is greater than the 'forecasted' quantity for a particular Forecast\_Entry. This occurs in one of two ways: (1) the 'committed' quantity is edited to be greater than the 'forecasted' quantity, or (2) the 'forecasted' is edited to be less than the 'committed' quantity. These situations violate the forecast limit imposed by the 'forecasted' field, and thus, produce this problem.

This problem is resolved by reducing the Forecast\_Entry 'committed' field to be no greater than the 'forecasted' field. This effectively honors the limit set by the 'forecasted' field. If no limit is desired, the 'forecasted' field should be set to "oo".

The OVER\_COMMITTED model has fields that references these models :

Extensions	OVER_CONSUMED Extension
------------	-------------------------

Forecast\_Entry.

forecast\_entry -- *a Forecast\_Entry field of model OVER\_COMMITTED*  
The Forecast\_Entry in which this problem occurred.  
Properties:   Export-Only Field

### OVER\_CONSUMED -- *a category extension of model Problem*

An OVER\_CONSUMED Problem indicates that the 'consumed' quantity is greater than the 'committed' quantity for a particular Forecast\_Entry. This occurs in one of two ways: (1) enough actual requests are promised to make the 'consumed' quantity greater than the 'committed' quantity, or (2) the 'committed' is edited to be less than the 'consumed' quantity.

This problem is resolved by increasing the Forecast\_Entry 'committed' field to be equal to the 'consumed' field.

The OVER\_CONSUMED model has fields that references these models :  
Forecast\_Entry.

forecast\_entry -- *a Forecast\_Entry field of model OVER\_CONSUMED*  
The Forecast\_Entry in which this problem occurred.  
Properties:   Export-Only Field

### UNALLOCATED\_FORECAST -- *a category extension of model Problem*

An UNALLOCATED\_FORECAST Problem indicates that 'date\_committed' is later than 'date\_allocated' for a particular Forecast\_Entry. This occurs in two ways: (1) the 'committed' is edited after the associated forecast requests have been planned and promised (i.e. allocated), or (2) the 'date\_committed' field was manually edited to a value later than the 'date\_allocated' field.

This problem is resolved by planning and promising the forecast request(s) associated with this problem's 'forecast\_entry'. This action causes the 'date\_allocated' field to be set later than the 'date\_committed' thereby eliminating this problem.

The UNALLOCATED\_FORECAST model has fields that references these models :  
Forecast\_Entry.

forecast\_entry -- *a Forecast\_Entry field of model UNALLOCATED\_FORECAST*  
The Forecast\_Entry in which this problem occurred.  
Properties:   Export-Only Field

Extensions	SUPPLY_PLANNED_LATE Extension
------------	-------------------------------

### SUPPLY\_PLANNED\_LATE -- *a category extension of model Problem*

A SUPPLY\_PLANNED\_LATE Problem indicates that the 'delivery\_plan' has been planned to satisfy the 'item\_request' but is in fact planned later than the 'receiving\_plan's start date. This Problem generally corresponds to a REQUEST\_PLANNED\_LATE Problem at the supplying Site.

To resolve this Problem, either resolve the REQUEST\_PLANNED\_LATE Problem at the supplying Site, or adjust the 'receiving\_plan' such that its start Date matches the supplying Site's 'delivery\_plan's end Date.

This is in both the SUPPLY and SUPPLY\_PLAN Problem\_Set categories, as well as SUPPLY\_PLANNED\_LATE.

The SUPPLY\_PLANNED\_LATE model has fields that references these models :  
Operation\_Plan, Item\_Request, Item\_Promise, Item\_Acceptance,  
Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance.

receiving\_plan -- *a Operation\_Plan field of model SUPPLY\_PLANNED\_LATE*  
This is the Operation\_Plan that will receive the delivery at this Site. Often, this is the Operation\_Plan that issued the 'item\_request'.  
Properties:   Export-Only Field

item\_request -- *a Item\_Request field of model SUPPLY\_PLANNED\_LATE*  
The Item\_Request that has this SUPPLY\_PLANNED\_LATE Problem. This 'item\_request.delivery\_plan' is planned to end later than this 'item\_request.owner.due' (the due Date of this Item\_Request).  
Properties:   Export-Only Field

item\_promise -- *a Item\_Promise field of model SUPPLY\_PLANNED\_LATE*  
The Item\_Promise for the 'item\_request'.  
Properties:   Export-Only Field

item\_acceptance -- *a Item\_Acceptance field of model SUPPLY\_PLANNED\_LATE*  
The Item\_Acceptance for the 'item\_request'.  
Properties:   Export-Only Field

Extensions	SUPPLY_PLANNED_EARLY Extension
------------	--------------------------------

delivery\_request - - a Delivery\_Request field of model  
SUPPLY\_PLANNED\_LATE  
This is simply shorthand for 'item\_request.owner'. The 'delivery\_plan' is planned to  
end later than this 'delivery\_request.due'.  
Properties:   Export-Only Field

delivery\_promise - - a Delivery\_Promise field of model  
SUPPLY\_PLANNED\_LATE  
This is simply shorthand for 'item\_promise.owner'.  
Properties:   Export-Only Field

delivery\_acceptance - - a Delivery\_Acceptance field of model  
SUPPLY\_PLANNED\_LATE  
This is simply shorthand for 'item\_acceptance.owner'.  
Properties:   Export-Only Field

request - - a Request field of model SUPPLY\_PLANNED\_LATE  
This is simply shorthand for 'item\_request.delivery\_request.owner'.  
Properties:   Export-Only Field

promise - - a Promise field of model SUPPLY\_PLANNED\_LATE  
This is simply shorthand for 'item\_promise.delivery\_promise.owner'.  
Properties:   Export-Only Field

acceptance - - a Acceptance field of model SUPPLY\_PLANNED\_LATE  
This is simply shorthand for 'item\_acceptance.delivery\_acceptance.owner'.  
Properties:   Export-Only Field

delivery\_plan - - a Operation\_Plan field of model SUPPLY\_PLANNED\_LATE  
This is simply shorthand for 'item\_request.delivery\_plan'. This 'delivery\_plan' is  
planned to end later than 'delivery\_request.due'.  
Properties:   Export-Only Field

SUPPLY\_PLANNED\_EARLY -- a category extension of model Problem

A SUPPLY\_PLANNED\_EARLY Problem indicates that the 'delivery\_plan' has been  
planned to satisfy the 'delivery\_request' but is in fact planned earlier than the  
'receiving\_plan's start Date. This Problem generally corresponds to a  
REQUEST\_PLANNED\_EARLY Problem at the supplying Site.

Extensions	SUPPLY_PLANNED_EARLY Extension
------------	--------------------------------

To resolve this Problem, either resolve the REQUEST\_PLANNED\_EARLY Problem  
at the supplying Site, or adjust the 'receiving\_plan' such that its start Date matches the  
supplying Site's 'delivery\_plan's end Date.

This is in both the SUPPLY and SUPPLY\_PLAN Problem\_Set categories, as well as  
SUPPLY\_PLANNED\_EARLY.

The SUPPLY\_PLANNED\_EARLY model has fields that references these models :  
Operation\_Plan, Item\_Request, Item\_Promise, Item\_Acceptance,  
Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance.

receiving\_plan - - a Operation\_Plan field of model SUPPLY\_PLANNED\_EARLY  
This is the Operation\_Plan that will receive the delivery at this Site. Often, this is the  
Operation\_Plan that issued the 'item\_request'.  
Properties:   Export-Only Field

item\_request - - a Item\_Request field of model SUPPLY\_PLANNED\_EARLY  
The Item\_Request that has this SUPPLY\_PLANNED\_EARLY Problem. This  
'item\_request.delivery\_plan' is planned to end earlier than this  
'item\_request.owner.due' (the due Date of this Item\_Request).  
Properties:   Export-Only Field

item\_promise - - a Item\_Promise field of model SUPPLY\_PLANNED\_EARLY  
The Item\_Promise for the 'item\_request'.  
Properties:   Export-Only Field

item\_acceptance - - a Item\_Acceptance field of model  
SUPPLY\_PLANNED\_EARLY  
The Item\_Acceptance for the 'item\_request'.  
Properties:   Export-Only Field

delivery\_request - - a Delivery\_Request field of model  
SUPPLY\_PLANNED\_EARLY  
This is simply shorthand for 'item\_request.owner'. The 'delivery\_plan' is planned to  
end earlier than this 'delivery\_request.due'.  
Properties:   Export-Only Field

delivery\_promise - - a Delivery\_Promise field of model  
SUPPLY\_PLANNED\_EARLY  
This is simply shorthand for 'item\_promise.owner'.  
Properties:   Export-Only Field

Extensions	SUPPLY_PLANNED_SHORT Extension
------------	--------------------------------

delivery\_acceptance - - a Delivery\_Acceptance field of model  
SUPPLY\_PLANNED\_EARLY  
This is simply shorthand for item\_acceptance.owner.  
Properties: Export-Only Field

request - - a Request field of model SUPPLY\_PLANNED\_EARLY  
This is simply shorthand for item\_request.delivery\_request.owner.  
Properties: Export-Only Field

promise - - a Promise field of model SUPPLY\_PLANNED\_EARLY  
This is simply shorthand for item\_promise.delivery\_promise.owner.  
Properties: Export-Only Field

acceptance - - a Acceptance field of model SUPPLY\_PLANNED\_EARLY  
This is simply shorthand for item\_acceptance.delivery\_acceptance.owner.  
Properties: Export-Only Field

delivery\_plan - - a Operation\_Plan field of model SUPPLY\_PLANNED\_EARLY  
This is simply shorthand for item\_request.delivery\_plan. This delivery\_plan is  
planned to end earlier than delivery\_request.due.  
Properties: Export-Only Field

#### SUPPLY\_PLANNED\_SHORT - - a category extension of model Problem

A SUPPLY\_PLANNED\_SHORT Problem indicates that the delivery\_plan has been  
planned to satisfy the item\_request but is in fact planned for less quantity than the  
receiving\_plan. This Problem generally corresponds to a  
REQUEST\_PLANNED\_SHORT Problem at the supplying Site.

To resolve this Problem, either resolve the REQUEST\_PLANNED\_SHORT Problem  
at the supplying Site, or reduce the receiving\_plan Quantity such that it matches the  
supplying Site's delivery\_plan's Quantity.

This is in both the SUPPLY and SUPPLY\_PLAN Problem\_Sets categories, as well as  
SUPPLY\_PLANNED\_SHORT.

The SUPPLY\_PLANNED\_SHORT model has fields that references these models :  
Operation\_Plan, Item\_Request, Item\_Promise, Item\_Acceptance,  
Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance.

Extensions	SUPPLY_PLANNED_SHORT Extension
------------	--------------------------------

receiving\_plan - - a Operation\_Plan field of model SUPPLY\_PLANNED\_SHORT  
This is the Operation\_Plan that will receive the delivery at this Site. Often, this is the  
Operation\_Plan that issued the item\_request.  
Properties: Export-Only Field

item\_request - - a Item\_Request field of model SUPPLY\_PLANNED\_SHORT  
The Item\_Request that has this SUPPLY\_PLANNED\_SHORT Problem. This  
item\_request.delivery\_plan is planned to deliver less quantity than this  
item\_request.quantity range.  
Properties: Export-Only Field

item\_promise - - a Item\_Promise field of model SUPPLY\_PLANNED\_SHORT  
The Item\_Promise for the item\_request.  
Properties: Export-Only Field

item\_acceptance - - a Item\_Acceptance field of model  
SUPPLY\_PLANNED\_SHORT  
The Item\_Acceptance for the item\_request.  
Properties: Export-Only Field

delivery\_request - - a Delivery\_Request field of model  
SUPPLY\_PLANNED\_SHORT  
This is simply shorthand for item\_request.owner.  
Properties: Export-Only Field

delivery\_promise - - a Delivery\_Promise field of model  
SUPPLY\_PLANNED\_SHORT  
This is simply shorthand for item\_promise.owner.  
Properties: Export-Only Field

delivery\_acceptance - - a Delivery\_Acceptance field of model  
SUPPLY\_PLANNED\_SHORT  
This is simply shorthand for item\_acceptance.owner.  
Properties: Export-Only Field

request - - a Request field of model SUPPLY\_PLANNED\_SHORT  
This is simply shorthand for item\_request.delivery\_request.owner.  
Properties: Export-Only Field

promise - - a Promise field of model SUPPLY\_PLANNED\_SHORT  
This is simply shorthand for item\_promise.delivery\_promise.owner.

Properties: Export-Only Field

acceptance - - *a Acceptance field of model SUPPLY\_PLANNED\_SHORT*

This is simply shorthand for item\_acceptance.delivery\_acceptance.owner.

Properties: Export-Only Field

delivery\_plan - - *a Operation\_Plan field of model SUPPLY\_PLANNED\_SHORT*

This is simply shorthand for item\_request.delivery\_plan. This delivery\_plan is planned to deliver less than item\_request.quantity.

Properties: Export-Only Field

SUPPLY\_PLANNED\_EXCESS - - *a category extension of model Problem*

A SUPPLY\_PLANNED\_EXCESS Problem indicates that the 'delivery\_plan' has been planned to satisfy the 'item\_request' but is in fact planned for more 'quantity' than the 'receiving\_plan'. This Problem generally corresponds to a REQUEST\_PLANNED\_EXCESS Problem at the supplying Site.

To resolve this Problem, either resolve the REQUEST\_PLANNED\_EXCESS Problem at the supplying Site, or increase the 'receiving\_plan' Quantity such that it matches the supplying Site's 'delivery\_plan's Quantity.

This is in both the SUPPLY and SUPPLY\_PLAN Problem\_Set categories, as well as SUPPLY\_PLANNED\_EXCESS.

The SUPPLY\_PLANNED\_EXCESS model has fields that references these models :  
Operation\_Plan, Item\_Request, Item\_Promise, Item\_Acceptance,  
Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance.

receiving\_plan - - *a Operation\_Plan field of model  
SUPPLY\_PLANNED\_EXCESS*

This is the Operation\_Plan that will receive the delivery at this Site. Often, this is the Operation\_Plan that issued the 'item\_request'.

Properties: Export-Only Field

item\_request - - *a Item\_Request field of model SUPPLY\_PLANNED\_EXCESS*

The Item\_Request that has this SUPPLY\_PLANNED\_EXCESS Problem. This 'item\_request.delivery\_plan' is planned to deliver more 'quantity' than this 'item\_request.quantity' range.

Properties: Export-Only Field

item\_promise - - *a Item\_Promise field of model SUPPLY\_PLANNED\_EXCESS*  
The Item\_Promise for the 'item\_request'.

Properties: Export-Only Field

item\_acceptance - - *a Item\_Acceptance field of model  
SUPPLY\_PLANNED\_EXCESS*

The Item\_Acceptance for the 'item\_request'.

Properties: Export-Only Field

delivery\_request - - *a Delivery\_Request field of model  
SUPPLY\_PLANNED\_EXCESS*

This is simply shorthand for item\_request.owner.

Properties: Export-Only Field

delivery\_promise - - *a Delivery\_Promise field of model  
SUPPLY\_PLANNED\_EXCESS*

This is simply shorthand for item\_promise.owner.

Properties: Export-Only Field

delivery\_acceptance - - *a Delivery\_Acceptance field of model  
SUPPLY\_PLANNED\_EXCESS*

This is simply shorthand for item\_acceptance.owner.

Properties: Export-Only Field

request - - *a Request field of model SUPPLY\_PLANNED\_EXCESS*  
This is simply shorthand for item\_request.delivery\_request.owner.

Properties: Export-Only Field

promise - - *a Promise field of model SUPPLY\_PLANNED\_EXCESS*  
This is simply shorthand for item\_promise.delivery\_promise.owner.

Properties: Export-Only Field

acceptance - - *a Acceptance field of model SUPPLY\_PLANNED\_EXCESS*  
This is simply shorthand for item\_acceptance.delivery\_acceptance.owner.

Properties: Export-Only Field

delivery\_plan - - *a Operation\_Plan field of model SUPPLY\_PLANNED\_EXCESS*

This is simply shorthand for 'item\_request.delivery\_plan'. This 'delivery\_plan' is planned to deliver more than 'item\_request.quantity'.

Properties: Export-Only Field

Extensions	SUPPLY_PROMISED_LATE Extension
SUPPLY_PROMISED_LATE -- a category extension of model Problem	
A SUPPLY_PROMISED_LATE Problem indicates that the Item_Promise has 'due' Dates that are later than the start Date of the 'receiving_plan'. The Promise is for later than requested. This Problem generally corresponds to a REQUEST_PROMISED_LATE Problem at the supplying Site.	
To resolve this Problem, either resolve the REQUEST_PROMISED_LATE Problem at the supplying Site, or adjust the 'receiving_plan' such that its start Date matches the supplying Site's 'delivery_promise's end Date.	
This is in the SUPPLY and SUPPLY_PROMISE Problem_Set categories, as well as SUPPLY_PROMISED_LATE.	
The SUPPLY_PROMISED_LATE model has fields that references these models : Operation_Plan, Item_Request, Item_Promise, Item_Acceptance, Delivery_Request, Delivery_Promise, Delivery_Acceptance.	
receiving_plan -- a Operation_Plan field of model SUPPLY_PROMISED_LATE This is the Operation_Plan that will receive the delivery at this Site. Often, this is the Operation_Plan that issued the Item_request. Properties: Export-Only Field	
Item_request -- a Item_Request field of model SUPPLY_PROMISED_LATE This is the Item_Request which was generated to request the supply Properties: Export-Only Field	
Item_Promise -- a Item_Promise field of model SUPPLY_PROMISED_LATE The Item_Promise for the Item_request. Properties: Export-Only Field	
Item_acceptance -- a Item_Acceptance field of model SUPPLY_PROMISED_LATE The Item_Acceptance for the Item_request. Properties: Export-Only Field	
delivery_request -- a Delivery_Request field of model SUPPLY_PROMISED_LATE The Delivery_Request whose 'delivery_promise' has later 'due' Dates than the receiving_plan Properties: Export-Only Field	

Extensions	SUPPLY_PROMISED_EARLY Extension
delivery_promise -- a Delivery_Promise field of model SUPPLY_PROMISED_LATE The Delivery_Promise whose 'due' Dates are later than its 'delivery_request'. Properties: Export-Only Field	
delivery_acceptance -- a Delivery_Acceptance field of model SUPPLY_PROMISED_LATE This is simply shorthand for 'Item_acceptance.owner'. Properties: Export-Only Field	
request -- a Request field of model SUPPLY_PROMISED_LATE This is simply shorthand for 'Item_request.delivery_request.owner'. Properties: Export-Only Field	
promise -- a Promise field of model SUPPLY_PROMISED_LATE This is simply shorthand for 'Item_promise.delivery_promise.owner'. Properties: Export-Only Field	
acceptance -- a Acceptance field of model SUPPLY_PROMISED_LATE This is simply shorthand for 'Item_acceptance.delivery_acceptance.owner'. Properties: Export-Only Field	
SUPPLY_PROMISED_EARLY -- a category extension of model Problem	
A SUPPLY_PROMISED_EARLY Problem indicates that the 'delivery_promise' has earlier 'due' Dates than the 'receiving_plan's start Date. The Promise is for earlier than requested. This Problem generally corresponds to a REQUEST_PROMISED_EARLY Problem at the supplying Site.	
To resolve this Problem, either resolve the REQUEST_PROMISED_EARLY Problem at the supplying Site, or adjust the 'receiving_plan' such that its start Date matches the supplying Site's 'delivery_promise's end Date.	
This is in the SUPPLY and SUPPLY_PROMISE Problem_Set categories, as well as SUPPLY_PROMISED_EARLY.	
The SUPPLY_PROMISED_EARLY model has fields that references these models : Operation_Plan, Item_Request, Item_Promise, Item_Acceptance, Delivery_Request, Delivery_Promise, Delivery_Acceptance.	

Extensions	SUPPLY_PROMISED_EARLY Extension
------------	---------------------------------

receiving\_plan - - a Operation\_Plan field of model  
SUPPLY\_PROMISED\_EARLY  
This is the Operation\_Plan that will receive the delivery at this Site. Often, this is the Operation\_Plan that issued the 'item\_request'.  
Properties: Export-Only Field

item\_request - - a Item\_Request field of model SUPPLY\_PROMISED\_EARLY  
This is the Item\_Request which was generated to request the supply  
Properties: Export-Only Field

item\_promise - - a Item\_Promise field of model SUPPLY\_PROMISED\_EARLY  
The Item\_Promise for the 'item\_request'.  
Properties: Export-Only Field

item\_acceptance - - a Item\_Acceptance field of model  
SUPPLY\_PROMISED\_EARLY  
The Item\_Acceptance for the 'item\_request'.  
Properties: Export-Only Field

delivery\_request - - a Delivery\_Request field of model  
SUPPLY\_PROMISED\_EARLY  
The Delivery\_Request whose 'delivery\_promise' has earlier 'due' Dates.  
Properties: Export-Only Field

delivery\_promise - - a Delivery\_Promise field of model  
SUPPLY\_PROMISED\_EARLY  
The Delivery\_Promise whose 'due' Dates are earlier than its 'delivery\_request'.  
Properties: Export-Only Field

delivery\_acceptance - - a Delivery\_Acceptance field of model  
SUPPLY\_PROMISED\_EARLY  
This is simply shorthand for 'item\_acceptance.owner'.  
Properties: Export-Only Field

request - - a Request field of model SUPPLY\_PROMISED\_EARLY  
This is simply shorthand for 'item\_request.delivery\_request.owner'.  
Properties: Export-Only Field

promise - - a Promise field of model SUPPLY\_PROMISED\_EARLY  
This is simply shorthand for 'item\_promise.delivery\_promise.owner'.  
Properties: Export-Only Field

Extensions	SUPPLY_PROMISED_SHORT Extension
------------	---------------------------------

acceptance - - a Acceptance field of model SUPPLY\_PROMISED\_EARLY  
This is simply shorthand for 'item\_acceptance.delivery\_acceptance.owner'.  
Properties: Export-Only Field

SUPPLY\_PROMISED\_SHORT - - a category extension of model Problem

A SUPPLY\_PROMISED\_SHORT Problem indicates that the 'item\_promise' has less 'quantity' than the 'receiving\_plan'. The Promise is for less than requested. This Problem generally corresponds to a REQUEST\_PROMISED\_SHORT Problem at the supplying Site.

To resolve this Problem, either resolve the REQUEST\_PROMISED\_SHORT Problem at the supplying Site, or reduce the 'receiving\_plan' Quantity such that it matches the supplying Site's 'delivery\_promise's Quantity.

This is in the SUPPLY and SUPPLY\_PROMISE Problem\_Sets categories, as well as SUPPLY\_PROMISED\_SHORT.

The SUPPLY\_PROMISED\_SHORT model has fields that references these models :  
Operation\_Plan, Item\_Request, Item\_Promise, Item\_Acceptance,  
Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance.

receiving\_plan - - a Operation\_Plan field of model  
SUPPLY\_PROMISED\_SHORT  
This is the Operation\_Plan that will receive the delivery at this Site. Often, this is the Operation\_Plan that issued the 'item\_request'.  
Properties: Export-Only Field

item\_request - - a Item\_Request field of model SUPPLY\_PROMISED\_SHORT  
The Item\_Request whose 'item\_promise' has less 'quantity'.  
Properties: Export-Only Field

item\_promise - - a Item\_Promise field of model SUPPLY\_PROMISED\_SHORT  
The Item\_Promise whose 'quantity' is less than its 'item\_request'.  
Properties: Export-Only Field

item\_acceptance - - a Item\_Acceptance field of model  
SUPPLY\_PROMISED\_SHORT  
The Item\_Acceptance for the 'item\_request'.  
Properties: Export-Only Field

**delivery\_request** - - *a Delivery\_Request field of model SUPPLY\_PROMISED\_SHORT*  
This is simply shorthand for 'item\_request.owner'.  
Properties: Export-Only Field

**delivery\_promise** - - *a Delivery\_Promise field of model SUPPLY\_PROMISED\_SHORT*  
This is simply shorthand for 'item\_promise.owner'.  
Properties: Export-Only Field

**delivery\_acceptance** - - *a Delivery\_Acceptance field of model SUPPLY\_PROMISED\_SHORT*  
This is simply shorthand for 'item\_acceptance.owner'.  
Properties: Export-Only Field

**request** - - *a Request field of model SUPPLY\_PROMISED\_SHORT*  
This is simply shorthand for 'item\_request.delivery\_request.owner'.  
Properties: Export-Only Field

**promise** - - *a Promise field of model SUPPLY\_PROMISED\_SHORT*  
This is simply shorthand for 'item\_promise.delivery\_promise.owner'.  
Properties: Export-Only Field

**acceptance** - - *a Acceptance field of model SUPPLY\_PROMISED\_SHORT*  
This is simply shorthand for 'item\_acceptance.delivery\_acceptance.owner'.  
Properties: Export-Only Field

**SUPPLY\_PROMISED\_EXCESS** - - *a category extension of model Problem*

A SUPPLY\_PROMISED\_EXCESS Problem indicates that the 'item\_promise' has more 'quantity' than its the 'receiving\_plan'. The Promise is for more than requested. This Problem generally corresponds to a REQUEST\_PROMISED\_EXCESS Problem at the supplying Site.

To resolve this Problem, either resolve the REQUEST\_PROMISED\_EXCESS Problem at the supplying Site, or increase the 'receiving\_plan' Quantity such that it matches the supplying Site's 'delivery\_promise's Quantity.

This is in the SUPPLY and SUPPLY\_PROMISE Problem\_Set categories, as well as SUPPLY\_PROMISED\_EXCESS.

The SUPPLY\_PROMISED\_EXCESS model has fields that reference these models :  
Operation\_Plan, Item\_Request, Item\_Promise, Item\_Acceptance,  
Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance.

**receiving\_plan** - - *a Operation\_Plan field of model SUPPLY\_PROMISED\_EXCESS*  
This is the Operation\_Plan that will receive the delivery at this Site. Often, this is the Operation\_Plan that issued the 'item\_request'.  
Properties: Export-Only Field

**item\_request** - - *a Item\_Request field of model SUPPLY\_PROMISED\_EXCESS*  
The Item\_Request whose 'item\_promise' has excess 'quantity'.  
Properties: Export-Only Field

**item\_promise** - - *a Item\_Promise field of model SUPPLY\_PROMISED\_EXCESS*  
The Item\_Promise whose 'quantity' is more than its 'item\_request'.  
Properties: Export-Only Field

**item\_acceptance** - - *a Item\_Acceptance field of model SUPPLY\_PROMISED\_EXCESS*  
The Item\_Acceptance for the 'item\_request'.  
Properties: Export-Only Field

**delivery\_request** - - *a Delivery\_Request field of model SUPPLY\_PROMISED\_EXCESS*  
This is simply shorthand for 'item\_request.owner'.  
Properties: Export-Only Field

**delivery\_promise** - - *a Delivery\_Promise field of model SUPPLY\_PROMISED\_EXCESS*  
This is simply shorthand for 'item\_promise.owner'.  
Properties: Export-Only Field

**delivery\_acceptance** - - *a Delivery\_Acceptance field of model SUPPLY\_PROMISED\_EXCESS*  
This is simply shorthand for 'item\_acceptance.owner'.  
Properties: Export-Only Field

**request** - - *a Request field of model SUPPLY\_PROMISED\_EXCESS*  
This is simply shorthand for 'item\_request.delivery\_request.owner'.  
Properties: Export-Only Field



Extensions	UNIDENTIFIED_OP_STATE Extension
------------	---------------------------------

Properties: Export-Only Field

**promise** - - - *a Promise field of model SUPPLY\_PROMISED\_EXCESS*

This is simply shorthand for item\_promise.delivery\_promise.owner.

Properties: Export-Only Field

**acceptance** - - - *a Acceptance field of model SUPPLY\_PROMISED\_EXCESS*

This is simply shorthand for item\_acceptance.delivery\_acceptance.owner.

Properties: Export-Only Field

**UNIDENTIFIED\_OP\_STATE** - - - *a category extension of model Problem*

An UNIDENTIFIED\_OP\_STATE problem indicates that the 'operation\_state' has not been able to identify the matching Op\_Plan to attached itself to.

To resolve this problem Operation\_State should identify an Op\_Plan using logic provided by it's identifier extension and attach itself to the selected Op\_Plan.

The UNIDENTIFIED\_OP\_STATE model has fields that references these models :  
Operation\_State.

operation\_state - - - *a Operation\_State field of model*

**UNIDENTIFIED\_OP\_STATE**

The Operation\_State that has this UNIDENTIFIED\_OP\_STATE problem.

Properties: Export-Only Field

**UNCONSOLIDATED** - - - *a category extension of model Problem*

Unconsolidated Operation\_Plans exist in this resource\_plan.

The UNCONSOLIDATED model has fields that references these models :

Resource\_Plan.

resource\_plan - - - *a Resource\_Plan field of model UNCONSOLIDATED*

The resource\_plan on which this problem is flagged

Properties: Export-Only Field

operation\_plans - - - *a List(Operation\_Plan) field of model UNCONSOLIDATED*

The list of operation\_plans

Properties: Export-Only Field

Extensions	UNCOORDINATED Extension
------------	-------------------------

**UNCOORDINATED** - - - *a category extension of model Problem*

The consolidation is not coordinated, i.e., all the operation\_plans in this consolidation do not start at the same time.

The UNCOORDINATED model has fields that references these models :

Resource\_Plan, Consolidation.

resource\_plan - - - *a Resource\_Plan field of model UNCOORDINATED*

The resource\_plan on which this problem is flagged

Properties: Export-Only Field

operation\_plans - - - *a List(Operation\_Plan) field of model UNCOORDINATED*

The list of operation\_plans

Properties: Export-Only Field

consolidation - - - *a Consolidation field of model UNCOORDINATED*

The consolidation that has this problem

Properties: Export-Only Field

**CONSOLIDATION\_OVERSIZE** - - - *a category extension of model Problem*

The Operation\_Plans in this Consolidation exceeds the max\_size of the Consolidation in one of the dimensions of its size

The CONSOLIDATION\_OVERSIZE model has fields that references these models :

Resource\_Plan, Consolidation.

resource\_plan - - - *a Resource\_Plan field of model*

**CONSOLIDATION\_OVERSIZE**

The resource\_plan on which this problem is flagged

Properties: Export-Only Field

operation\_plans - - - *a List(Operation\_Plan) field of model*

**CONSOLIDATION\_OVERSIZE**

The list of operation\_plans

Properties: Export-Only Field

consolidation - - - *a Consolidation field of model CONSOLIDATION\_OVERSIZE*

The consolidation that has this problem

Properties: Export-Only Field

oversize\_dimensions - - *a List(Symbol) field of model CONSOLIDATION\_OVERSIZE*  
The dimensions on which oversize occurred  
Properties: Export-Only Field

oversize (Symbol) - - *a Percentage field of model CONSOLIDATION\_OVERSIZE*  
The percentage oversize in a dimension 'name'  
Properties: Export-Only Field

**CONSOLIDATION\_UNDERSIZE** -- *a category extension of model Problem*

The Operation\_Plan in this Consolidation consume less than the min\_size of the Consolidation in one of the dimensions of its size

The CONSOLIDATION\_UNDERSIZE model has fields that references these models :  
Resource\_Plan, Consolidation.

resource\_plan - - *a Resource\_Plan field of model CONSOLIDATION\_UNDERSIZE*  
The resource\_plan on which this problem is flagged  
Properties: Export-Only Field

operation\_plans - - *a List(Operation\_Plan) field of model CONSOLIDATION\_UNDERSIZE*  
The list of operation\_plans  
Properties: Export-Only Field

consolidation - - *a Consolidation field of model CONSOLIDATION\_UNDERSIZE*  
The consolidation that has this problem  
Properties: Export-Only Field

undersize\_dimensions - - *a List(Symbol) field of model CONSOLIDATION\_UNDERSIZE*  
The dimensions on which undersize occurred  
Properties: Export-Only Field

undersize (Symbol) - - *a Percentage field of model CONSOLIDATION\_UNDERSIZE*  
The percentage oversize in a dimension 'name'

Properties: Export-Only Field

**OVERLOAD** -- *a category extension of model Problem*

An OVERLOAD Problem indicates that the load planned during these 'dates' on the 'resource\_plan' is greater than the capacity during these 'dates'. The 'excess\_load\_time' gives the severity of the OVERLOAD in actual hours, and the 'excess\_load\_std\_time' gives the severity of the OVERLOAD in standard hours.

The OVERLOAD model has fields that references these models :  
Resource\_Plan.

resource\_plan - - *a Resource\_Plan field of model OVERLOAD*  
The Resource\_Plan with this OVERLOAD Problem.  
Properties: Export-Only Field

overload\_time - - *a Time field of model OVERLOAD*  
The excess 'load\_time' during the 'dates' of this Problem. This is 'resource\_plan.load\_time(dates)' - resource\_plan.available\_time(dates)'.  
Properties: Export-Only Field

overload\_std\_time - - *a Time field of model OVERLOAD*  
The excess 'load\_std\_time' during the 'dates' of this Problem. This is 'resource\_plan.load\_std\_time(dates)' - resource\_plan.capacity(dates)'.  
Properties: Export-Only Field

**OVERSIZE** -- *a category extension of model Problem*

An OVERSIZE Problem indicates that the load planned during these 'dates' on the 'resource\_plan' has size greater than available.

The OVERSIZE model has fields that references these models :  
Resource\_Plan.

resource\_plan - - *a Resource\_Plan field of model OVERSIZE*  
The Resource\_Plan with this OVERSIZE Problem.  
Properties: Export-Only Field

**oversize - - a Percentage field of model OVERSIZE**

The loads on the 'resource\_plan' during these 'dates' exceed the 'size' of the Resource by this Percentage. A value of 0% indicates no excess (no Problem). A value of 100% indicates that the peak size usage during 'dates' is twice the available 'size'. Note that a 'size' extension could be considering multiple dimensions of 'size'; this can indicate the maximum oversize of any dimension; or if the dimensions can naturally be combined (volume from 3 lengths), it can indicate the combined figure.

Properties: Export-Only Field

**BUCKET\_OVERSIZE - - a category extension of model Problem**

This problem is defined for SHARED\_USE Load\_Policy only.

A BUCKET\_OVERSIZE Problem indicates that the load planned during a time bucket on the 'resource\_plan' is greater than 'max\_bucket\_load', specified in the SHARED\_USE load\_policy of the resource. The problem period is the same as the bucket period.

The BUCKET\_OVERSIZE model has fields that references these models :  
Resource\_Plan.

resource\_plan - - a Resource\_Plan field of model BUCKET\_OVERSIZE  
The Resource\_Plan with this BUCKET\_OVERSIZE Problem.  
Properties: Export-Only Field

bucket\_oversize - - a Percentage field of model BUCKET\_OVERSIZE  
(load\_size/available\_capacity - max\_bucket\_load) during the problem period.  
available\_capacity is the size of the resource in the problem period. load\_size is the sum of contribution of all the load\_plans in this period  
Properties: Export-Only Field

**UNDERLOAD - - a category extension of model Problem**

This problem is defined for SHARED\_USE Load\_Policy only.

An UNDERLOAD Problem indicates that the load planned during the problem period on the 'resource\_plan' is less than desired. The minimum desired load on this resource is specified as 'min\_load' in the SHARED\_USE Load\_Policy. The problem period begins at 'Plan\_start' and ends at 'min\_load\_fence' specified in the SHARED\_USE Load\_Policy.

Be careful with this Problem. Utilizing Resources just to utilize them is usually bad practice. However, there are times when it is known that a Resource must stay utilized (it is a fluctuating bottleneck), and it is useful to work ahead to handle unexpected variation.

The UNDERLOAD model has fields that references these models :

Resource\_Plan.

resource\_plan - - a Resource\_Plan field of model UNDERLOAD  
The Resource\_Plan with this UNDERLOAD Problem.

Properties: Export-Only Field

underload - - a Percentage field of model UNDERLOAD  
(min\_load - (load\_size/available\_capacity)) during the problem period. This is the percentage of the resource capacity that is not utilized. available\_capacity is calculated from the size of the resource in the problem period. load\_size is the sum of contribution of all the load\_plans in this period  
Properties: Export-Only Field

**NEGATIVE\_ON\_HAND - - a category extension of model Problem**

A NEGATIVE\_ON\_HAND Problem indicates that the flow planned during these 'dates' on the 'buffer\_plan' results in the 'on\_hand' being less than zero. Such a Problem is inherently infeasible.

The NEGATIVE\_ON\_HAND model has fields that references these models :  
Buffer\_Plan.

buffer\_plan - - a Buffer\_Plan field of model NEGATIVE\_ON\_HAND  
The Buffer\_Plan with this NEGATIVE\_ON\_HAND Problem.  
Properties: Export-Only Field

low\_on\_hand - - a Quantity field of model NEGATIVE\_ON\_HAND  
The lowest (most negative) value of 'buffer\_plan.on\_hand' during these 'dates'. This Quantity is converted to the 'unit' of the Buffer.  
Properties: Export-Only Field

deficit - - a Quantity field of model NEGATIVE\_ON\_HAND  
The additional Quantity needed during these 'dates' to reach an acceptable 'on\_hand' level. This Quantity is converted to the 'unit' of the Buffer.  
Properties: Export-Only Field

**OVER\_FLOW\_LIMIT -- a category extension of model Problem**

A OVER\_FLOW\_LIMIT Problem indicates that the flow planned during these 'dates' on the 'buffer\_plan' results in the 'on\_hand' being less than zero. These problems are identical to NEGATIVE\_ON\_HAND problems except that these problems are generated by the FLOW\_LIMIT\_CALENDAR flow policy. This flow policy is to be used in modelling resources (unit capacity buffers).

The OVER\_FLOW\_LIMIT model has fields that references these models :  
Buffer\_Plan.

buffer\_plan -- a Buffer\_Plan field of model OVER\_FLOW\_LIMIT  
The Buffer\_Plan with this OVER\_FLOW\_LIMIT Problem.

Properties: Export-Only Field

**low\_on\_hand -- a Quantity field of model OVER\_FLOW\_LIMIT**

The lowest (most negative) value of 'buffer\_plan.on\_hand' during these 'dates'. This Quantity is converted to the unit of the Buffer.

Properties: Export-Only Field

**deficit -- a Quantity field of model OVER\_FLOW\_LIMIT**

The additional Quantity needed during these 'dates' to reach an acceptable 'on\_hand' level. This Quantity is converted to the unit of the Buffer.

Properties: Export-Only Field

**NEGATIVE\_ON\_HAND\_AT\_END -- a category extension of model Problem**

A NEGATIVE\_ON\_HAND\_AT\_END Problem indicates that the flow planned on the 'buffer\_plan' results in the 'on\_hand' being less than zero after the end of the planning horizon. This special case of NEGATIVE\_ON\_HAND is often used to propagate demand upstream. Such a Problem is inherently infeasible.

The NEGATIVE\_ON\_HAND\_AT\_END model has fields that references these models :  
Buffer\_Plan.

buffer\_plan -- a Buffer\_Plan field of model NEGATIVE\_ON\_HAND\_AT\_END

The Buffer\_Plan with this NEGATIVE\_ON\_HAND Problem.

Properties: Export-Only Field

low\_on\_hand -- a Quantity field of model NEGATIVE\_ON\_HAND\_AT\_END  
The lowest (most negative) value of 'buffer\_plan.on\_hand' during these 'dates'. This Quantity is converted to the unit of the Buffer.

Properties: Export-Only Field

**deficit -- a Quantity field of model NEGATIVE\_ON\_HAND\_AT\_END**

The additional Quantity needed during these 'dates' to reach an acceptable 'on\_hand' level. This Quantity is converted to the unit of the Buffer.

Properties: Export-Only Field

**LOT\_OVER\_CONSUMED -- a category extension of model Problem**

A LOT\_OVER\_CONSUMED Problem indicates that supplying flows planned for a specific Lot are insufficient for the consuming flows of that Lot. The supplying flows may be smaller than the consuming flows, or they may be later. This is similar to a NEGATIVE\_ON\_HAND Problem, but is used instead in Lot-for-Lot Buffers to indicate the specific Lot assignment that has the problem.

The LOT\_OVER\_CONSUMED model has fields that references these models :  
Buffer\_Plan.

buffer\_plan -- a Buffer\_Plan field of model LOT\_OVER\_CONSUMED

The Buffer\_Plan with this LOW\_ON\_HAND Problem.

Properties: Export-Only Field

**shortage\_quantity -- a Quantity field of model LOT\_OVER\_CONSUMED**

The quantity difference between the supplying Lots and the consuming Lot. For example, if a 70 unit supplier is paired with a 100 unit consumer, this value will be -30 units. Note that this value could be zero (or even positive) if shortage\_time is not zero.

Properties: Export-Only Field

**shortage\_time -- a Time field of model LOT\_OVER\_CONSUMED**

If a Lot is planned earlier than the supplying Lots it consumes from, this is the amount of time between the planned start date of the consumer and the planned completion date of the earliest Lot it consumes from. If the consuming Lot is planned as late as, or later than, all the Lots it consumes from, this field will be zero. This could happen if 'shortage\_quantity' is less than zero.

Properties: Export-Only Field

**LOT\_NOT\_CONSUMED -- a category extension of model Problem**

A LOT\_NOT\_CONSUMED Problem indicates that supplying flows planned for a specific Lot are not allocated to any consuming flows.

The LOT\_NOT\_CONSUMED model has fields that references these models :  
Buffer\_Plan.

buffer\_plan - - a Buffer\_Plan field of model LOT\_NOT\_CONSUMED  
The Buffer\_Plan with this LOT\_NOT\_CONSUMED Problem.  
Properties: Export-Only Field

LOT\_NOT\_PRODUCED -- a category extension of model Problem

A LOT\_NOT\_PRODUCED Problem indicates that consuming flows planned for a Buffer are not allocated to any producing flows.

The LOT\_NOT\_PRODUCED model has fields that references these models :  
Buffer\_Plan.

buffer\_plan - - a Buffer\_Plan field of model LOT\_NOT\_PRODUCED  
The Buffer\_Plan with this LOT\_NOT\_CONSUMED Problem.  
Properties: Export-Only Field

LOT\_OVER\_PRODUCED -- a category extension of model Problem

A LOT\_OVER\_PRODUCED Problem indicates that supplying flows planned for a specific Lot are excessive for the consuming flows of that Lot. The supplying flows may be larger than the consuming flows, or they may be earlier. This is similar to an EXCESS\_ON\_HAND Problem, but is used instead in Lot-for-Lot Buffers to indicate the specific Lot assignment that has the problem.

The LOT\_OVER\_PRODUCED model has fields that references these models :  
Buffer\_Plan.

buffer\_plan - - a Buffer\_Plan field of model LOT\_OVER\_PRODUCED  
The Buffer\_Plan with this LOW\_ON\_HAND Problem.  
Properties: Export-Only Field

excess\_quantity - - a Quantity field of model LOT\_OVER\_PRODUCED  
The quantity difference between the producing Lots and the consuming Lot. For example, if a 70 unit producer is paired with a 50 unit consumer, this value will be 20 units. Note that this value could be zero (or even negative) if excess\_time is not zero.  
Properties: Export-Only Field

excess\_time - - a Time field of model LOT\_OVER\_PRODUCED  
If a Lot is planned later than the supplying Lots it consumes from, this is the amount of time between the planned start date of the consumer and the planned completion date of the earliest Lot it consumes from. If the consuming Lot is planned as early as, or earlier than, all the Lots it consumes from, this field will be zero. This could happen if 'excess\_quantity' is greater than zero.  
Properties: Export-Only Field

LOW\_ON\_HAND -- a category extension of model Problem

A LOW\_ON\_HAND Problem indicates that the flow planned during these 'dates' on the 'buffer\_plan' results in the 'on\_hand' being less than 'min\_on\_hand' or other on\_hand limits. Such Problems are typically feasible, but undesirable for various reasons (mostly safety stock and service level related reasons).

The LOW\_ON\_HAND model has fields that references these models :  
Buffer\_Plan.

buffer\_plan - - a Buffer\_Plan field of model LOW\_ON\_HAND  
The Buffer\_Plan with this LOW\_ON\_HAND Problem.  
Properties: Export-Only Field

low\_on\_hand - - a Quantity field of model LOW\_ON\_HAND  
The lowest value of 'buffer\_plan.on\_hand' during these 'dates'. This Quantity is converted to the unit of the Buffer.  
Properties: Export-Only Field

deficit - - a Quantity field of model LOW\_ON\_HAND  
The additional Quantity needed during these 'dates' to reach an acceptable on\_hand level. This Quantity is converted to the unit of the Buffer.  
Properties: Export-Only Field

EXCESS\_ON\_HAND -- a category extension of model Problem

An EXCESS\_ON\_HAND Problem indicates that the flow planned during these 'dates' on the 'buffer\_plan' results in the 'on\_hand' being in excess of desired levels. Such a Problem is typically feasible, but undesirable for various reasons (to prevent excess inventory and all the related costs).

The EXCESS\_ON\_HAND model has fields that references these models :  
Buffer\_Plan.

buffer\_plan -- a Buffer\_Plan field of model EXCESS\_ON\_HAND  
The Buffer\_Plan with this EXCESS\_ON\_HAND Problem.  
Properties: Export-Only Field

excess -- a Quantity field of model EXCESS\_ON\_HAND  
The largest excess Quantity that is in buffer\_plan.on\_hand during these dates. Note that this is the Quantity beyond the desired levels and beyond the permissible excess levels. This Quantity is converted to the unit of the Buffer.  
Properties: Export-Only Field

EXCESS\_ON\_HAND\_AT\_END -- a category extension of model Problem

An EXCESS\_ON\_HAND\_AT\_END Problem indicates that the flow planned on the buffer\_plan results in the on\_hand being in excess of desired levels at the end of the planning horizon. Such a Problem is typically feasible, but undesirable for various reasons (to prevent excess inventory and all the related costs).

The EXCESS\_ON\_HAND\_AT\_END model has fields that references these models :  
Buffer\_Plan.

buffer\_plan -- a Buffer\_Plan field of model EXCESS\_ON\_HAND\_AT\_END  
The Buffer\_Plan with this EXCESS\_ON\_HAND\_AT\_END Problem.  
Properties: Export-Only Field

excess -- a Quantity field of model EXCESS\_ON\_HAND\_AT\_END  
The excess Quantity that is in buffer\_plan at the end of the planning horizon. Note that this is the Quantity beyond the desired levels and beyond the permissible excess levels. This Quantity is converted to the unit of the Buffer.  
Properties: Export-Only Field

10.23 Active\_Strategy Extensions

10.23.1 termination extensions of model Active\_Strategy

NO\_PROBLEMS -- a termination extension of model Active\_Strategy

A NO\_PROBLEMS termination Active\_Strategy will run until it finds no improvements to the plan for more than 'max\_stable\_time' or until it has run in all for more than 'max\_run\_time'. It specifies no additional termination condition.

TARGET\_ACHIEVED -- a termination extension of model Active\_Strategy

A TARGET\_ACHIEVED termination Active\_Strategy will run until it achieves the target as specified by the 'goal' extension fields. Note that some 'goal' extensions specify no target, and thus will never be terminated by achieving the target.

RESOLVE\_COUNT\_EXCEEDED -- a termination extension of model Active\_Strategy

A RESOLVE\_COUNT\_EXCEEDED termination Strategy will run until it has attempted to resolve 'max\_resolve\_count' problems.

MANUAL -- a termination extension of model Active\_Strategy

A MANUAL termination Strategy will run until user hits the 'done' button (sets 'done' to "true"). This is generally used as a sub\_strategy to pause for the user to make some decisions. As an Active\_Strategy, it can present to the user the relevant Problems and Goal measures. Pressing 'done' ends that sub\_strategy, and continues on with the next.

done -- a Logical field of model MANUAL  
This Active\_Strategy will terminate if 'done' is set "true".  
Default: false

10.23.2 execution extensions of model Active\_Strategy

SEQUENCE\_RUN\_ONCE -- a execution extension of model Active\_Strategy

A SEQUENCE\_RUN\_ONCE strategy executes its 'sub\_strategies' in a particular order. Each Ordered\_Sub\_Strategy will be run exactly once till it terminated. The order is defined using 'sequence' field of Ordered\_Sub\_Strategy model. The lower the number the earlier the strategy is performed.

sub\_strategies -- a List(Active\_Strategy) field of model  
**SEQUENCE\_RUN\_ONCE**  
the list of Active\_Strategy that will be performed sequentially  
Properties: Export-Only Field

current\_sub -- a Number field of model **SEQUENCE\_RUN\_ONCE**  
current sub\_strategy that is running  
Properties: Export-Only Field

**SEQUENCE\_RUN\_MULTIPLE** -- a execution extension of model Active\_Strategy

A **SEQUENCE\_RUN\_MULTIPLE** strategy executes its 'sub\_strategies' in a particular order. Each Ordered\_Sub\_Strategy will be run exactly once till it terminated. The order is defined using 'sequence' field of Ordered\_Sub\_Strategy model. The lower the number the earlier the strategy is performed.

sub\_strategies -- a List(Active\_Strategy) field of model  
**SEQUENCE\_RUN\_MULTIPLE**  
the list of Active\_Strategy that will be performed sequentially  
Properties: Export-Only Field

current\_sub -- a Number field of model **SEQUENCE\_RUN\_MULTIPLE**  
current sub\_strategy that is running  
Properties: Export-Only Field

**BEFORE\_AND\_AFTER** -- a execution extension of model Active\_Strategy

See extension **BEFORE\_AND\_AFTER** for Strategy

The **BEFORE\_AND\_AFTER** model has fields that references these models :  
**Active\_Strategy**.

before\_run -- a Active\_Strategy field of model **BEFORE\_AND\_AFTER**  
This Strategy will run before the start of 'owner' Strategy run.  
Default: [unspecified]  
Properties: Export-Only Field

before\_search -- a Active\_Strategy field of model **BEFORE\_AND\_AFTER**  
This strategy will run before the start of each search.  
Default: [unspecified]  
Properties: Export-Only Field

after\_resolve -- a Active\_Strategy field of model **BEFORE\_AND\_AFTER**  
This strategy will run after each problem is resolved.  
Default: [unspecified]  
Properties: Export-Only Field

after\_search -- a Active\_Strategy field of model **BEFORE\_AND\_AFTER**  
This strategy will run after each search is complete.  
Default: [unspecified]  
Properties: Export-Only Field

after\_success -- a Active\_Strategy field of model **BEFORE\_AND\_AFTER**  
This Strategy will run after each successful search. It will only run if a search results in a better Plan, where better is defined by the Strategy 'goals', then the search is termed a "success".  
Default: [unspecified]  
Properties: Export-Only Field

after\_run -- a Active\_Strategy field of model **BEFORE\_AND\_AFTER**  
This Strategy will run at the completion of 'owner' Strategy run.  
Default: [unspecified]  
Properties: Export-Only Field

**SEQUENTIAL\_ALTERNATES** -- a execution extension of model Active\_Strategy

See extension **SEQUENTIAL\_ALTERNATES** for Strategy

The **SEQUENTIAL\_ALTERNATES** model has fields that references these models :  
**Active\_Strategy**.

propagation -- a Active\_Strategy field of model **SEQUENTIAL\_ALTERNATES**  
This Strategy will run after each resolve of the parent.  
Default: [unspecified]  
Properties: Export-Only Field

evaluation -- a Active\_Strategy field of model **SEQUENTIAL\_ALTERNATES**  
The goal function of this strategy is used to evaluate each alternate.  
Default: [unspecified]  
Properties: Export-Only Field

cleanup -- a Active\_Strategy field of model **SEQUENTIAL\_ALTERNATES**  
This strategy will run after each failed alternate, before moving to the next alternate.

Extensions	SEQUENTIAL_ALTERNATES_KEEP_BEST Extension
------------	---

Default: [unspecified]  
 Properties: Export-Only Field

**min\_alt** -- *a Integer field of model SEQUENTIAL\_ALTERNATES*  
 The min\_alt and max\_alt defines the range of alternate operations considered. For example, if min\_alt = max\_alt = 0, only the first alternate is allowed. The min and max are duplicated in Active\_Execution because of the possibility of recursive strategy, where each subsequent invocation may have different values.  
 Default: 0

**max\_alt** -- *a Integer field of model SEQUENTIAL\_ALTERNATES*  
 See notes in min\_alt.  
 Default: 100

**current\_alt** -- *a Integer field of model SEQUENTIAL\_ALTERNATES*  
 The current alternate allowed  
 Properties: Export-Only Field

**SEQUENTIAL\_ALTERNATES\_KEEP\_BEST** -- *a execution extension of model Active\_Strategy*

See extension SEQUENTIAL\_ALTERNATES\_KEEP\_BEST for Strategy

The SEQUENTIAL\_ALTERNATES\_KEEP\_BEST model has fields that references these models :  
 Active\_Strategy.

**propagation** -- *a Active\_Strategy field of model SEQUENTIAL\_ALTERNATES\_KEEP\_BEST*  
 This Strategy will run after each resolve of the parent.  
 Default: [unspecified]  
 Properties: Export-Only Field

**evaluation** -- *a Active\_Strategy field of model SEQUENTIAL\_ALTERNATES\_KEEP\_BEST*  
 The goal function of this strategy is used to evaluate each alternate.  
 Default: [unspecified]  
 Properties: Export-Only Field

Extensions	PROPORTIONAL_RESOLVES Extension
------------	---------------------------------

**min\_alt** -- *a Integer field of model SEQUENTIAL\_ALTERNATES\_KEEP\_BEST*  
 The min\_alt and max\_alt defines the range of alternate operations considered. For example, if min\_alt = max\_alt = 0, only the first alternate is allowed. The min and max are duplicated in Active\_Execution because of the possibility of recursive strategy, where each subsequent invocation may have different values.  
 Default: 0

**max\_alt** -- *a Integer field of model SEQUENTIAL\_ALTERNATES\_KEEP\_BEST*  
 See notes in min\_alt.  
 Default: 100

**PROPORTIONAL\_RESOLVES** -- *a execution extension of model Active\_Strategy*

See extension PROPORTIONAL\_RESOLVES for Strategy  
*sub\_strategies* -- *a List(Active\_Strategy) field of model PROPORTIONAL\_RESOLVES*  
 the list of Active\_Strategy that will be performed sequentially  
 Properties: Export-Only Field

**ORDERED\_RESOLVES** -- *a execution extension of model Active\_Strategy*

A Ordered\_Resolve strategy executes the highest rank Ranked\_Sub\_Strategy with a problem after each resolve.

**sub\_strategies** -- *a List(Active\_Strategy) field of model ORDERED\_RESOLVES*  
 A List of Active\_Strategy that will be executed during the run of owner Active\_Strategy.  
 Properties: Export-Only Field

10.23.3 problem\_selection extensions of model Active\_Strategy

**PROPORTIONAL\_INTERACTION** -- *a problem\_selection extension of model Active\_Strategy*

The PROPORTIONAL\_INTERACTION problem\_selection extension selects the problem to solve using following logic.

1. It first selects the Problem\_Set probabilistically, weighted by Problem\_Set.focus times the total interaction of problems within that Problem\_Set. 2. Choose a problem within selected Problem\_Set probabilistically, weighted by the interaction value of Problem. Otherwise return the current chosen problem.



Extensions	EARLIEST_PROBLEM_START Extension
------------	----------------------------------

At Strategy.heat == 0, it will first select the Problem\_Set with the largest Problem\_Set.focus value and within that it will select the Problem with the largest interaction value.

When extension's behavior becomes deterministic, it will break the ties by selecting the problem with the earliest startdate. If there are multiple problems with the same start date, it will break the further ties by selecting problem randomly.

#### EARLIEST\_PROBLEM\_START -- a problem\_selection extension of model Active\_Strategy

The EARLIEST\_PROBLEM\_START problem\_selection extension selects the problem probabilistically, weighted by problem's start date. As the Strategy.heat approaches zero, probability of selecting the problem with the earliest start date increases continuously. At Strategy.selection.heat == 0, it will be completely deterministic and will always select the problem with the earliest start date first.

When the selection becomes deterministic, it will break the tie among the same start date problems, by selecting the problem with the highest interaction. If there are multiple problems with the same interaction value, it will break the further tie by selecting a problem randomly.

#### SORT\_BY\_EXPRESSION -- a problem\_selection extension of model Active\_Strategy

The SORT\_BY\_EXPRESSION problem\_selection extension selects the first problem from the sorted problem list. The problem list is sorted using the user specified 'sort\_criteria' expression.

Example Usage:

sort\_criteria

Extensions	Active_Goal Extensions
------------	------------------------

### 10.24 Active\_Goal Extensions

#### 10.24.1 goal extensions of model Active\_Goal

##### FEASIBILITY -- a goal extension of model Active\_Goal

The goal is to minimize the sum of 'interaction' of infeasible Problems, ultimately driving it to zero by eliminating all infeasible Problems.

total\_interaction -- a Number field of model FEASIBILITY  
The current level for the sum of 'interaction' of all infeasible Problems. If this value is less than or equal to 'strategy\_goal.target\_interaction', then the target has been met (though the goal to minimize it to zero remains).

This is typically used in conjunction with target-oriented 'termination' extensions.  
Properties: Export-Only Field

##### MINIMIZE\_PROBLEM\_COUNT -- a goal extension of model Active\_Goal

The goal is to minimize the number of problems ultimately driving it to zero by eliminating all Problems.

problem\_count -- a Number field of model MINIMIZE\_PROBLEM\_COUNT  
The current number of problems. If this value is less than or equal to 'strategy\_goal.problem\_count', then the target has been met (though the goal to minimize it to zero remains).

This is typically used in conjunction with target-oriented 'termination' extensions.  
Properties: Export-Only Field

##### MINIMIZE\_PROBLEMS -- a goal extension of model Active\_Goal

The goal is to minimize the sum of the weights of all problems, ultimately driving it to zero by eliminating all Problems.

problems -- a Number field of model MINIMIZE\_PROBLEMS  
The sum of the weights of the current problems. If this value is less than or equal to 'strategy\_goal.problems', then the target has been met (though the goal to minimize it to zero remains).

This is typically used in conjunction with target-oriented 'termination' extensions.  
Default: 0

Extensions	MINIMIZE_LATENESS Extension
------------	-----------------------------

Properties:    Export-Only Field

**MINIMIZE\_LATENESS -- a goal extension of model Active\_Goal**

The goal is to minimize the sum of 'lateness' of all requests.

*total\_lateness* - - a Number field of model MINIMIZE\_LATENESS

The current level for the sum of 'lateness' of all requests. If this value is less than or equal to 'strategy\_goal.target\_lateness', then the target has been met (though the goal to minimize it to zero remains).

This is typically used in conjunction with target-oriented 'termination' extensions.

Properties:    Export-Only Field

**WEIGHTED\_LATENESS -- a goal extension of model Active\_Goal**

The goal is to minimize the weighted sum of 'lateness' of all requests.

*total\_lateness* - - a Number field of model WEIGHTED\_LATENESS

The current level for the sum of 'lateness' of all requests. If this value is less than or equal to 'strategy\_goal.target\_lateness', then the target has been met (though the goal to minimize it to zero remains).

This is typically used in conjunction with target-oriented 'termination' extensions.

Properties:    Export-Only Field

**WEIGHTED\_SHORTNESS -- a goal extension of model Active\_Goal**

The goal is to minimize the weighted sum of 'lateness' of all requests.

*total\_shortness* - - a Number field of model WEIGHTED\_SHORTNESS

The current level for the sum of 'shortness' of all requests. If this value is less than or equal to 'strategy\_goal.target\_shortness', then the target has been met (though the goal to minimize it to zero remains).

This is typically used in conjunction with target-oriented 'termination' extensions.

Properties:    Export-Only Field

**MINIMIZE\_SHORTNESS -- a goal extension of model Active\_Goal**

The goal is to minimize the sum of 'shortness' of all requests

Extensions	MINIMIZE_COST Extension
------------	-------------------------

*total\_shortness* - - a Number field of model MINIMIZE\_SHORTNESS

The current level for the sum of 'shortness' of all requests. If this value is less than or equal to 'strategy\_goal.target\_shortness', then the target has been met (though the goal to minimize it to zero remains).

This is typically used in conjunction with target-oriented 'termination' extensions.

Properties:    Export-Only Field

**MINIMIZE\_COST -- a goal extension of model Active\_Goal**

The goal is to minimize the cost of the current plan

*total\_cost* - - a Number field of model MINIMIZE\_COST

The current level for the 'cost' of the plan. If this value is less than or equal to 'strategy\_goal.target\_cost', then the target has been met (though the goal to minimize it to zero remains).

This is typically used in conjunction with target-oriented 'termination' extensions.

Default:    0

Properties:    Export-Only Field

**MAXIMIZE\_PROFIT -- a goal extension of model Active\_Goal**

The goal is to maximize 'profit' (the difference between 'revenue' and 'cost').

*total\_profit* - - a Number field of model MAXIMIZE\_PROFIT

The current level for 'profit' (the difference between 'revenue' and 'cost'). If this value is greater than or equal to 'strategy\_goal.target\_profit', then the target has been met (though the goal to maximize it remains).

This is typically used in conjunction with target-oriented 'termination' extensions.

Default:    0

Properties:    Export-Only Field

**MAXIMIZE\_REVENUE -- a goal extension of model Active\_Goal**

The goal is to maximize 'revenue'.

*total\_revenue* - - a Number field of model MAXIMIZE\_REVENUE

The current level for 'revenue'. If this value is greater than or equal to 'strategy\_goal.target\_revenue', then the target has been met (though the goal to maximize it remains).

<i>Extensions</i>	<i>MAXIMIZE REVENUE Extension</i>
-------------------	-----------------------------------

This is typically used in conjunction with target-oriented 'termination' extensions.  
 Default: 0  
 Properties: Export-Only Field

<i>Extensions</i>	<i>Item Extensions</i>
-------------------	------------------------

## 10.25 Item Extensions

### 10.25.1 spec extensions of model Item

#### STANDARD -- a spec extension of model Item

All Configurations of this Item are interchangeable -- fully equivalent for planning purposes. No additional fields are needed in order to specify the Lot to find or build.

#### CUSTOM -- a spec extension of model Item

The Item is specified as an order-specific item, such as is the norm in engineer-to-order businesses. In these businesses, the items are designed for each customer. However, modeling a new Item for each customer order would be pointless. Thus, general families of items are modeled with Items, and the order-specific items are simply the Configurations produced. Thus, no two Configurations of this Item will be interchangeable.

Note that no Lot-specific information is maintained. However, 'lots\_tracked' will be "true" because they are each assumed distinct.

Extensions	Skill Extensions
------------	------------------

### 10.26 Skill Extensions

#### 10.26.1 selection extensions of model Skill

**PRIMARY** -- *a selection extension of model Skill*

The 'primary' Resource is selected by default. The others can be chosen manually.

The PRIMARY model has fields that references these models :  
Resource.

**primary** - - *a Resource field of model PRIMARY*  
The primary Resource that is selected by default.  
Default: [unspecified]

**PREFER\_PRIMARY** -- *a selection extension of model Skill*

The 'primary' Resource is selected by default. The others could be chosen while planning, when moving to an alternate resource. These other alternate resources are chosen at random. While we chose an alternate resource, if the load is already on an alternate resource (a resource other than the primary), then this always chooses the primary. The alternate resources can also be chosen manually.

The PREFER\_PRIMARY model has fields that references these models :  
Resource.

**primary** - - *a Resource field of model PREFER\_PRIMARY*  
The primary Resource that is selected by default.  
Default: [unspecified]

**EVEN** -- *a selection extension of model Skill*

The 'resources' are selected with even probability. Note that this is the default for Skills generated for each Resource Family.

**MAX\_EFFICIENCY** -- *a selection extension of model Skill*

The 'resources' are selected in order of maximum 'efficiency'.

Extensions	Skill_Resource Extensions
------------	---------------------------

### 10.27 Skill\_Resource Extensions

#### 10.27.1 efficiency extensions of model Skill\_Resource

**FIXED** -- *a efficiency extension of model Skill\_Resource*

A FIXED efficiency Skill\_Resource has a fixed efficiency level at all times. The default value is the very common case of 100%.

**fixed\_efficiency** - - *a Percentage field of model FIXED*  
The efficiency level of the resource in performing this skill  
Default: 100%

**CALENDAR** -- *a efficiency extension of model Skill\_Resource*

A CALENDAR efficiency Skill\_Resource has efficiency that will vary over time as specified in a Calendar.

Further, the options for increasing efficiency by incurring additional cost can be modeled. Calendar\_Entry's with a non-zero 'charge' are treated as options for increasing efficiency when needed. Uses the Default\_Efficiency\_Calendar, if efficiency\_calendar is not set.

The CALENDAR model has fields that references these models :  
Calendar.

**efficiency\_calendar** - - *a Calendar field of model CALENDAR*  
A Calendar with PERCENTAGE entries', where the Percentage is the efficiency level at those dates. Entries with a non-zero 'charge' are treated as options for increasing efficiency when needed.  
Default: [unspecified]

Extensions	Configuration Extensions
------------	--------------------------

### 10.28 Configuration Extensions

#### 10.28.1 spec extensions of model Configuration

##### STANDARD -- a spec extension of model Configuration

All Configurations of this Item are interchangeable -- fully equivalent for planning purposes. No additional fields are needed in order to specify the Lot to find or build.

##### CUSTOM -- a spec extension of model Configuration

The Item is specified as an order-specific item, such as is the norm in engineer-to-order businesses. In these businesses, the items are designed for each customer. However, modeling a new Item for each customer order would be pointless. Thus, general families of items are modeled with Items, and the order-specific items are simply the Configurations produced. Thus, no two Configurations of this Item will be interchangeable.

Note that no Lot-specific information is maintained. However, 'lots\_tracked' will be "true" because they are each assumed distinct.

Extensions	Operation_State Extensions
------------	----------------------------

### 10.29 Operation\_State Extensions

#### 10.29.1 identifier extensions of model Operation\_State

##### EARLIEST -- a identifier extension of model Operation\_State

Identifies the earliest non-released Op\_Plan for the 'operation' and attaches the State information to it.

The EARLIEST model has fields that references these models :  
Operation.

release\_name -- a Symbol field of model EARLIEST  
The release name of the identified Op\_Plan.  
Default: [unspecified]

operation -- a Operation field of model EARLIEST  
Operation for which the state information is reported.  
Default: [unspecified]

top\_operation -- a Operation field of model EARLIEST  
top most operation in the 'operation' tree.  
Default: [unspecified]

#### 10.29.2 state\_spec extensions of model Operation\_State

##### STARTED -- a state\_spec extension of model Operation\_State

A STARTED Operation\_State's 'quantity' specifies the cumulative Quantity that has been started by the identified Operation\_Plan as of 'date'.

Note that this is a cumulative Quantity, it does not shrink as material is completed. The quantity being processed is 'completed + scrapped - started'. This Operation\_State also implies the ARRIVED Quantity is at least as large.

The STARTED model has fields that references these models :  
Item.

item -- a Item field of model STARTED  
The Item for which the starting 'quantity' is reported.  
Default: [unspecified]

Extensions	COMPLETED Extension
------------	---------------------

quantity -- *a Quantity field of model STARTED*  
The Quantity that has started execution (production, transportation, etc.), as of 'date'.  
If the Item is unspecified, then this Quantity needs to be unitless and it represents the units of operation plan.  
Default: 0

COMPLETED -- *a state\_spec extension of model Operation\_State*

A COMPLETED Operation\_State's 'quantity' specifies the cumulative Quantity that has been completed by the identified Operation\_Plan as of 'date'.

Note that this is a cumulative Quantity -- the total ever completed by this Operation\_State. This Operation\_State also implies that the ARRIVED and STARTED Quantity's are at least as large as the sum of the SCRAPED and COMPLETED Quantity's.

The COMPLETED model has fields that references these models :  
Item.

Item -- *a Item field of model COMPLETED*  
The Item for which the completed 'quantity' is reported.  
Default: [unspecified]

quantity -- *a Quantity field of model COMPLETED*  
The Quantity that has completed execution (production, transportation, etc.), as of 'date'.

If the Item is unspecified, then this Quantity needs to be unitless and it represents the units of operation plan.  
Default: 0

IN\_FRONT -- *a state\_spec extension of model Operation\_State*

An IN\_FRONT Operation\_State's 'quantity' specifies the Quantity of this Operation that is "in front" of the Operation, ready to start. It is the Quantity that has 'arrived' but not yet 'started'.

Note that IN\_FRONT state information can only be meaningful if the full snapshot of the super\_operation\_plan is given. Therefore, IN\_FRONT Operation\_State's are ignored unless their 'date' is equal to the latest of all Operation\_States of all Operation\_Plans under the top\_operation\_plan.

Extensions	IN_FRONT Extension
------------	--------------------

quantity -- *a Quantity field of model IN\_FRONT*  
The Quantity of this Operation that is standing "in front" of this Operation -- the Quantity that has 'arrived' but not 'started'.  
Default: 0

<i>Extensions</i>	<i>Request Extensions</i>
-------------------	---------------------------

10.30 Request Extensions

10.30.1 delivery\_policy extensions of model Request

**FIXED** -- a delivery\_policy extension of model Request

Fixed delivery policy

10.30.2 delivery\_naming extensions of model Request

**NUMBERED** -- a delivery\_naming extension of model Request

Delivery\_Requests are named by successive numbers.

<i>Extensions</i>	<i>Delivery_Request Extensions</i>
-------------------	------------------------------------

10.31 Delivery\_Request Extensions

10.31.1 promising\_policy extensions of model Delivery\_Request

**BUCKETED\_ALL** -- a promising\_policy extension of model Delivery\_Request

This is similar to ALL promising policy, but in addition to that it takes care of consume\_earlier, Generic\_Products, Alternate\_Products, and the Generic Products that are shared between the forecasts of the items of a delivery request. It is similar to Bucketed\_All\_Min\_Price except that this policy would give quotes from all the matching forecasts.

**BUCKETED\_ASAP** -- a promising\_policy extension of model Delivery\_Request

This is similar to ASAP promising policy, but in addition to that it takes care of consume\_earlier, Generic\_Products, Alternate\_Products, and the Generic Products that are shared between the forecasts of the items of a delivery request. It is similar to Bucketed\_All\_Min\_Price\_ASAP except that this policy would give quotes from all the matching forecasts.

**ON\_TIME** -- a promising\_policy extension of model Delivery\_Request

If an ON\_TIME promise\_policy Delivery\_Request cannot be satisfied, then the promise should only be made for the amount/price that can be delivered on the date requested (it will be promised short). The Delivery\_Request should not be promised late.

**ALL** -- a promising\_policy extension of model Delivery\_Request

If an ALL promising\_policy Delivery\_Request cannot be satisfied, then the promise should only be made for the entire amount/price (it will be promised late). The Delivery\_Request should not be promised short. This policy will not work for cases of Product consume\_earlier set to anything other than the default value.

**ALL\_ON\_TIME** -- a promising\_policy extension of model Delivery\_Request

If an ALL\_ON\_TIME promising\_policy Delivery\_Request cannot be satisfied for the entire amount/price on the date requested, then it should not be promised. The Delivery\_Request should not be promised short or late. This policy will not work for cases of Product consume\_earlier set to anything other than the default value.

**ASAP** -- a promising\_policy extension of model Delivery\_Request

If an ASAP promising\_policy Delivery\_Request cannot be satisfied, then it should be promised so as to deliver as much of the request as possible as soon as possible. This may split the request into a number of deliveries. This policy will not work for cases of Product consume\_earlier set to anything other than the default value.

**ASAP\_MONTHLY -- a promising\_policy extension of model Delivery\_Request**

If an ASAP\_MONTHLY promising\_policy Delivery\_Request cannot be satisfied, then it should be promised so as to deliver as much of the request as possible on the date requested and the rest will split into monthly deliveries. The monthly deliveries will be as late in the month as required to obtain the maximum amount of the available to promise for that month, but no later. This policy will not work for cases of Product consume\_earlier set to anything other than the default value.

**BUCKETED\_ALLOCATION -- a promising\_policy extension of model Delivery\_Request**

If a BUCKETED\_ALLOCATION promising\_policy Delivery\_Request cannot be satisfied on the date requested, then it should be promised as to deliver as much of the request as possible on the date requested. The remaining promises will split into bucketed deliveries where the buckets correspond to the forecast horizon. The EARLY bucketed deliveries will each be as late in the bucket as required to obtain the maximum amount of product available in that bucket. For promises in LATE buckets (as well as late promises in the requested bucket), the deliveries will be according to ASAP. That is, there will be one delivery for each date where there is additional quantity available for at least one item request. Preference is given to ON TIME promises first, EARLY promises next, and finally LATE promises. NOTE: Assumes that Product consume\_earlier settings do not span multiple forecast horizon buckets.

**BUCKETED\_ALL\_MIN\_PRICE -- a promising\_policy extension of model Delivery\_Request**

This policy's behavior would be similar to that of ALL with some exceptions. One of them currently is that the promising\_policy\_atp would quote a single product only for a given 'item\_request'. If it finds that all the item requests could be satisfied by a date, then that date would be the delivery date of this policy's delivery\_atp. If or a given item request, the product that gives all the quantities by the delivery date would be considered. If more than one product could satisfy by the delivery date, then one of them with the minimum 'price' would be chosen.

This policy gives only one delivery\_atp. This helps to know the 'delivery\_date' of all the 'quantity's of all the item requests that are being delivered together. Furthermore, it would contain one item\_atp for each item\_request. Each item\_atp would contain multiple product\_atp's. The product\_atp's 'dates' would be different from each other depending on the atp\_entries dates. But when the consumption takes place using this policy, then multiple delivery\_atp's and item\_atp's would be created due to and according to the different dates of the product\_atp's.

This policy works well with generic products. The incremental quotes given by the policy considers the cumulative quantities available on each atp\_entry for the generic products. This policy works well with consume\_earlier. The generic products and the specific products could set consume\_earlier and expect the policy to give individual quotes considering the cumulative quantities available in those ATP\_Entry periods. When there are multiple item\_requests in a delivery\_request, then the delivery date of this policy's promising\_policy\_atp would be that of the latest date of all the 'product\_atp's combined. Note that the product\_atp's would still have their own 'dates' depending on the buckets in which those quantities were found. The offered quantities would be such that the consumption would be as close to the delivery date as possible. Finally, if not all the item\_requests could be satisfied within the planning horizon, then this policy would not give any quote.

**BUCKETED\_MIN\_PRICE\_ASAP -- a promising\_policy extension of model Delivery\_Request**

This policy's behavior would be similar to that of BUCKETED\_ALL\_MIN\_PRICE with some differences. This policy gives promises such that as much of delivery as possible could be made on the due date, and the rest of the quantities could be delivered as soon as possible, if all of the quantities could not be satisfied by the due date. There would be multiple deliveries starting from the due date, if all the requested quantities could not be delivered on the requested date. For a given item request, the product that could deliver the quantities earliest would be considered. If more than one product could satisfy by the same date, then one of them with the minimum 'price' would be chosen. This policy works well with generic products. The incremental quotes given by the policy considers the cumulative quantities available on each atp\_entry for the generic products. If you use consume\_earlier, then early deliveries could occur. That behavior could change in the later versions of this promising policy.



Extensions	SHIP_IN_RATIO Extension
------------	-------------------------

**SHIP\_IN\_RATIO -- a promising\_policy extension of model Delivery\_Request**

This is similar to ASAP promising policy, except that each delivery would contain all the items requested in the ratio of the quantities of the original delivery request. This policy takes care of generic products, consume\_earlier, generic and alternate products shared among the products of the items in the item requests. When such sharing occur, this policy would try to split the available quantity among the items in such a way that the ratio of the items in the delivery could be maintained.

**10.31.2 fulfillment\_policy extensions of model Delivery\_Request**

**ON\_TIME -- a fulfillment\_policy extension of model Delivery\_Request**

If any item in an ON\_TIME Delivery\_Request can not be delivered on time, then the delivery will be shorted. That is, a partial delivery will be made rather than waiting until a full delivery can be made.

**FULL\_QUANTITIES\_OF\_ALL\_ITEMS -- a fulfillment\_policy extension of model Delivery\_Request**

If any item in a FULL\_QUANTITIES\_OF\_ALL\_ITEMS Delivery\_Request is unavailable on the delivery date, then the whole delivery will be delayed rather than making a partial delivery.

**UNRESTRICTED -- a fulfillment\_policy extension of model Delivery\_Request**

An UNRESTRICTED Delivery\_Request can be shorted, delayed, or both.

Extensions	Promise Extensions
------------	--------------------

**10.32 Promise Extensions**

**10.32.1 delivery\_policy extensions of model Promise**

**FIXED -- a delivery\_policy extension of model Promise**

Fixed delivery policy

<i>Extensions</i>	<i>Delivery_Promise Extensions</i>
-------------------	------------------------------------

10.33 Delivery\_Promise Extensions

10.33.1 fulfillment\_policy extensions of model Delivery\_Promise

ON\_TIME -- *a fulfillment\_policy extension of model Delivery\_Promise*

If any item in an ON\_TIME Delivery\_Promise can not be delivered on time, then the delivery will be shorted. That is, a partial delivery will be made rather than waiting until a full delivery can be made.

FULL\_QUANTITIES\_OF\_ALL\_ITEMS -- *a fulfillment\_policy extension of model Delivery\_Promise*

If any item in a FULL\_QUANTITIES\_OF\_ALL\_ITEMS Delivery\_Promise is unavailable on the delivery date, then the whole delivery will be delayed rather than making a partial delivery.

UNRESTRICTED -- *a fulfillment\_policy extension of model Delivery\_Promise*

An UNRESTRICTED Delivery\_Promise can be shorted, delayed, or both.

<i>Extensions</i>	<i>Horizon Extensions</i>
-------------------	---------------------------

10.34 Horizon Extensions

10.34.1 bucket\_spec extensions of model Horizon

ONE -- *a bucket\_spec extension of model Horizon*

The ONE bucket\_spec Horizon will contain a single bucket for the entire duration.

MONTHS -- *a bucket\_spec extension of model Horizon*

The horizon will be broken up on calendar month boundaries.

WEEKS -- *a bucket\_spec extension of model Horizon*

The horizon will be broken up into week buckets followed by buckets on calendar month boundaries.

first\_day\_of\_week -- *a Integer field of model WEEKS*  
Monday is day 1; Sunday is day 7.

Default: 1

week\_buckets -- *a Number field of model WEEKS*

The number of week buckets. After that number of week buckets, monthly buckets begin. If 00, then the entire horizon is broken into weeks.

Default: 00

DAYS -- *a bucket\_spec extension of model Horizon*

A DAYS Horizon will be broken up into day buckets followed by buckets on calendar month boundaries.

day\_buckets -- *a Number field of model DAYS*

The number of day buckets. After that number of day buckets, monthly buckets begin.

If 00, then the entire horizon is broken into days.

Default: 00

DATES -- *a bucket\_spec extension of model Horizon*

Extensions	DATES Extension
------------	-----------------

The bucket\_starts are maintained in sorted order, and there can be no duplicates. When generating buckets, all Dates that are outside the specified Date\_Range are forgotten. And the specified Date\_Range 'start' and 'end' are added to the front and rear of the List (unless already present). Thus, the first bucket will always start at the 'start' of the specified Date\_Range and the last bucket will always end at the 'end' of the specified Date\_Range.

Each bucket will be a Date\_Range that starts at the specified 'start' Date and ends at the next specified 'start' Date, which is the start of the next bucket. (A Date\_Range includes all Dates up to, but not including its 'end' Date -- thus, consecutive buckets will leave no gaps and will have no overlaps.)

This extension provides lots of flexibility to user in specifying any arbitrary bucket sizes.

The DATES model has these submodels :

Horizon\_Bucket\_Start.

The DATES model has fields that references these models :

Horizon\_Bucket\_Start.

bucket\_starts -- a list of Horizon\_Bucket\_Start submodels of model DATES

List of bucket\_starts specifying bucket boundaries.

Extensions	Delivery_Acceptance Extensions
------------	--------------------------------

### 10.35 Delivery\_Acceptance Extensions

10.35.1 fulfillment\_policy extensions of model Delivery\_Acceptance

ON\_TIME -- a fulfillment\_policy extension of model Delivery\_Acceptance

If any item in an ON\_TIME Delivery\_Acceptance can not be delivered on time, then the delivery will be shorted. That is, a partial delivery will be made rather than waiting until a full delivery can be made.

FULL\_QUANTITIES\_OF\_ALL\_ITEMS -- a fulfillment\_policy extension of model Delivery\_Acceptance

If any item in a FULL\_QUANTITIES\_OF\_ALL\_ITEMS Delivery\_Acceptance is unavailable on the delivery date, then the whole delivery will be delayed rather than making a partial delivery.

UNRESTRICTED -- a fulfillment\_policy extension of model Delivery\_Acceptance

An UNRESTRICTED Delivery\_Acceptance can be shorted, delayed, or both.

10.36 Strategy Extensions

10.36.1 termination extensions of model Strategy

NO\_PROBLEMS -- a termination extension of model Strategy

A NO\_PROBLEMS termination Strategy will run until there are no more problems or it finds no improvements to the plan for more than 'max\_stable\_time' or until it has run in all for more than 'max\_run\_time'. It specifies no additional termination condition.

problem\_filter - - a Expression field of model NO\_PROBLEMS

An Expression that is passed a Problem as '#' and should return "true" if the Problem should be counted.

Default: true

TARGET\_ACHIEVED -- a termination extension of model Strategy

A TARGET\_ACHIEVED termination Strategy will run until it achieves the target as specified by the 'goal' extension fields. Note that some 'goal' extensions specify no target, and thus will never be terminated by achieving the target.

RESOLVE\_COUNT\_EXCEEDED -- a termination extension of model Strategy

A RESOLVE\_COUNT\_EXCEEDED termination Strategy will run until it has attempted to resolve 'max\_resolve\_count' problems.

MANUAL -- a termination extension of model Strategy

A MANUAL termination Strategy will run until user hit done button

10.36.2 execution extensions of model Strategy

SEQUENCE\_RUN\_ONCE -- a execution extension of model Strategy

A SEQUENCE\_RUN\_ONCE strategy executes it's 'sub\_strategies' in a particular sequence once. Each Ordered\_Sub\_Strategy will be run to completion once in a given order. Sequence is defined using 'sequence' field of Ordered\_Sub\_Strategy model. The lower the number the earlier the strategy is performed.

The SEQUENCE\_RUN\_ONCE model has these submodels :  
Ordered\_Sub\_Strategy.

The SEQUENCE\_RUN\_ONCE model has fields that references these models :

Ordered\_Sub\_Strategy.

sub\_strategies - - a list of Ordered\_Sub\_Strategy submodels of model SEQUENCE\_RUN\_ONCE

An Ordered\_Sub\_Strategy that will be performed in a given sequence.

SEQUENCE\_RUN\_MULTIPLE -- a execution extension of model Strategy

A SEQUENCE\_RUN\_MULTIPLE strategy executes it's 'sub\_strategies' in a particular sequence. It repeats that sequence until its termination condition is reached. Each Ordered\_Sub\_Strategy will be run to completion. Sequence is defined using 'sequence' field of Ordered\_Sub\_Strategy model. The lower the number the earlier the strategy is performed.

The SEQUENCE\_RUN\_MULTIPLE model has these submodels :  
Ordered\_Sub\_Strategy.

The SEQUENCE\_RUN\_MULTIPLE model has fields that references these models :  
Ordered\_Sub\_Strategy.

sub\_strategies - - a list of Ordered\_Sub\_Strategy submodels of model SEQUENCE\_RUN\_MULTIPLE

An Ordered\_Sub\_Strategy that will be performed in a given sequence.

BEFORE\_AND\_AFTER -- a execution extension of model Strategy

A BEFORE\_AND\_AFTER strategy allows execution of different Strategy's before and after 'run', 'search' and 'resolve' stages of it's owner Strategy run.

A Strategy "run" continues until one of it's termination conditions are met. A "run" consists of a series of "searches". A "search" is a series of Problem "resolves" that result in a new plan. If a search results in a better Plan, where better is defined by the Strategy 'goals', then the search is termed a "success".

This extension can be used in many different ways. User can setup a 'after\_run/search' Strategy that will drive the plan towards feasibility at the end of each 'search' or 'run' or setup a cleanup strategy that will do certain cleanup tasks(i.e. get rid of excess) after each 'search' or 'run'.

The 'after\_resolve' strategy will run after each problem is resolved.

The BEFORE\_AND\_AFTER model has fields that references these models :

Strategy.

**before\_run** - - *a Strategy field of model BEFORE\_AND\_AFTER*

This Strategy will run before the start of 'owner' Strategy run.

Default: [unspecified]

**before\_search** - - *a Strategy field of model BEFORE\_AND\_AFTER*

This strategy will run before the start of each search.

Default: [unspecified]

**after\_resolve** - - *a Strategy field of model BEFORE\_AND\_AFTER*

This strategy will run after each problem is resolved.

Default: [unspecified]

**after\_search** - - *a Strategy field of model BEFORE\_AND\_AFTER*

This strategy will run after each search is complete.

Default: [unspecified]

**after\_success** - - *a Strategy field of model BEFORE\_AND\_AFTER*

This Strategy will run after each successful search. It will only run if a search results in a better Plan, where better is defined by the Strategy 'goals', then the search is termed a "success".

Default: [unspecified]

**after\_run** - - *a Strategy field of model BEFORE\_AND\_AFTER*

This Strategy will run at the completion of 'owner' Strategy run.

Default: [unspecified]

**SEQUENTIAL\_ALTERNATES** - - *a execution extension of model Strategy*

A SEQUENTIAL\_ALTERNATES strategy allows the user to search through a series of alternates. A SEQUENTIAL\_ALTERNATES strategy chooses a problem and then iterates from min\_alt rank to max\_alt rank. For each rank the resolver is called (restricted by the change category to move to or resize on only alternates with that rank). Then the propagation sub-strategy is called. If the result achieves the goal of the evaluation strategy then the loop is terminated. Else the the cleanup sub-strategy is run and we move to the next alternate.

The SEQUENTIAL\_ALTERNATES model has fields that references these models : Strategy.

**propagation** - - *a Strategy field of model SEQUENTIAL\_ALTERNATES*

This Strategy will run after each resolve of the parent.

Default: [unspecified]

**evaluation** - - *a Strategy field of model SEQUENTIAL\_ALTERNATES*

The goal function of this strategy is used to evaluate each alternate.

Default: [unspecified]

**cleanup** - - *a Strategy field of model SEQUENTIAL\_ALTERNATES*

This strategy will run after each failed alternate, before moving to the next alternate.

Default: [unspecified]

**min\_alt** - - *a Integer field of model SEQUENTIAL\_ALTERNATES*

The min\_alt and max\_alt defines the range of alternate operations considered. For example, if min\_alt = max\_alt = 0, only the first alternate is allowed.

Default: 0

**max\_alt** - - *a Integer field of model SEQUENTIAL\_ALTERNATES*

See notes in min\_alt.

Default: 100

**SEQUENTIAL\_ALTERNATES\_KEEP\_BEST** - - *a execution extension of model Strategy*

A SEQUENTIAL\_ALTERNATES\_KEEP\_BEST strategy allows the user to search through a series of alternates and keep the best move under a given

chooses a problem and then iterates from min\_alt rank to max\_alt rank. For each rank the resolver is called (restricted by the change category to move to or resize on only alternates with that rank). Then the propagation sub-strategy is called. If the result achieves the goal of the evaluation strategy then the loop is terminated. Else the strategy undoes to the point before the resolver call and moves to the next alternate. Once all alternates have been examined, the strategy returns to the best alternate (as measured by the goal of the evaluation strategy), calls the resolver restricted to that alternate, and then runs the propagation strategy.

The SEQUENTIAL\_ALTERNATES\_KEEP\_BEST model has fields that references these models : Strategy.

**propagation** - - *a Strategy field of model*

**SEQUENTIAL\_ALTERNATES\_KEEP\_BEST**

This Strategy will run after each resolve of the parent.

Default: [unspecified]

evaluation -- a Strategy field of model

SEQUENTIAL\_ALTERNATES\_KEEP\_BEST

The goal function of this strategy is used to evaluate each alternate.

Default: [unspecified]

min\_alt -- a Integer field of model SEQUENTIAL\_ALTERNATES\_KEEP\_BEST

The min\_alt and max\_alt defines the range of alternate operations considered. For example, if min\_alt = max\_alt = 0, only the first alternate is allowed.

Default: 0

max\_alt -- a Integer field of model

SEQUENTIAL\_ALTERNATES\_KEEP\_BEST

See notes in min\_alt.

Default: 100

PROPORTIONAL\_RESOLVES -- a execution extension of model Strategy

A PROPORTIONAL\_RESOLVES strategy executes it's 'ranked\_sub\_strategies'. It probabilistically selects a Ranked\_Sub\_Strategy to perform one problem resolution.

The criteria of probabilistic selection is: Ranked\_Sub\_Strategy.rank \*

Ranked\_Sub\_Strategy.strategy.total\_interaction

The PROPORTIONAL\_RESOLVES model has these submodels :

Ranked\_Sub\_Strategy.

The PROPORTIONAL\_RESOLVES model has fields that references these models :

Ranked\_Sub\_Strategy.

ranked\_sub\_strategies -- a list of Ranked\_Sub\_Strategy submodels of model PROPORTIONAL\_RESOLVES

A Ranked\_Sub\_Strategy that can be selected and performed by the 'owner' Strategy.

ORDERED\_RESOLVES -- a execution extension of model Strategy

A Ordered\_Resolve strategy executes the highest rank Ranked\_Sub\_Strategy with a problem after each resolve.

The ORDERED\_RESOLVES model has these submodels :

Ordered\_Sub\_Strategy.

The ORDERED\_RESOLVES model has fields that references these models :  
Ordered\_Sub\_Strategy.

sub\_strategies -- a list of Ordered\_Sub\_Strategy submodels of model

ORDERED\_RESOLVES

A Ranked\_Sub\_Strategy that can be selected and performed by the 'owner' Strategy.

10.36.3 problem\_selection extensions of model Strategy

PROPORTIONAL\_INTERACTION -- a problem\_selection extension of model Strategy

The PROPORTIONAL\_INTERACTION problem\_selection extension selects the problem to solve using following logic.

1. It first selects the Problem\_Set probabilistically, weighted by Problem\_Set.focus times the total interaction of problems within that Problem\_Set. 2. Choose a problem within selected Problem\_Set probabilistically, weighted by the interaction value of Problem. Otherwise return the current chosen problem.

At Strategy.heal == 0, it will first select the Problem\_Set with the largest Problem\_Set.focus value and within that it will select the Problem with the largest interaction value.

When extension's behavior becomes deterministic, it will break the ties by selecting the problem with the earliest startdate. If there are multiple problems with the same start date, it will break the further ties by selecting problem randomly.

EARLIEST\_PROBLEM\_START -- a problem\_selection extension of model Strategy

The EARLIEST\_PROBLEM\_START problem\_selection extension selects the problem probabilistically, weighted by problem's start date. As the Strategy.heal approaches zero, probability of selecting the problem with the earliest start date increases continuously. At Strategy.selection\_heal == 0, it will be completely deterministic and will always select the problem with the earliest\_start date first.

When the selection becomes deterministic, it will break the tie among the same start date problems, by selecting the problem with the highest interaction. If there are multiple problems with the same interaction value, it will break the further tie by selecting a problem randomly.

SORT\_BY\_EXPRESSION -- a problem\_selection extension of model Strategy

Extensions	<i>SORT_BY_EXPRESSION</i> Extension
------------	-------------------------------------

The *SORT\_BY\_EXPRESSION* problem\_selection extension selects the first problem from the sorted problem list. The problem list is sorted using the user specified *sort\_criteria* expression that is passed two *Active\_Problems* "a" & "b".

Example Usage:

1) *strategy.sorting\_criteria* = "a.problem.dates.start < b.problem.dates.start" The above expression will sort the problem list in an ascending order of problem start dates and this extension will pick the earliest start date problem first.

2) *strategy.sorting\_criteria* = "if (a.problem.dates.start != b.problem.dates.start, a.problem.dates.start < b.problem.dates.start, a.interaction > b.interaction) The above expression specifies Problem.start date as a primary key and *Active\_Problem.interaction* as a secondary key.

Note, if no *sorting\_criteria* expression is specified, problem list will be sorted by problem creation date.

*sorting\_criteria* - - *a* Expression field of model *SORT\_BY\_EXPRESSION*  
 An Expression that is passed two *Active\_Problems* "a" & "b" If "a" should be selected before "b", return true, otherwise false.  
 Default: true

Extensions	<i>Problem_Set</i> Extensions
------------	-------------------------------

### 10.37 Problem\_Set Extensions

10.37.1 category extensions of model *Problem\_Set*

*REQUEST\_NOT\_PLANNED* -- *a* category extension of model *Problem\_Set*

A *REQUEST\_NOT\_PLANNED* Problem\_Set selects *REQUEST\_NOT\_PLANNED* Problems with certain characteristics.

This is a subset of the *REQUEST* Problem\_Set category.

*item\_request\_filter* - - *a* Expression field of model *REQUEST\_NOT\_PLANNED*  
 An Expression that is passed an *Item\_Request* as "i" and should return "true" if the *REQUEST\_NOT\_PLANNED* Problems with that *Item\_Request* should be included in the *Problem\_Set*. If it returns "false", then *REQUEST\_NOT\_PLANNED* Problems with that *Item\_Request* will be excluded.  
 Default: true

*problem\_filter* - - *a* Expression field of model *REQUEST\_NOT\_PLANNED*  
 An Expression that is passed a *Problem* as "p" and should return "true" if the identified *REQUEST\_NOT\_PLANNED* Problems should be included in the *Problem\_Set*. If it returns "false", then the identified Problems will be excluded.  
 Default: true

*REQUEST\_PLANNED\_LATE* -- *a* category extension of model *Problem\_Set*

A *REQUEST\_PLANNED\_LATE* Problem\_Set selects *REQUEST\_PLANNED\_LATE* Problems with certain characteristics.

This is a subset of the *REQUEST* and *REQUEST\_PLAN* Problem\_Set categories.

*min\_lateness* - - *a* Time field of model *REQUEST\_PLANNED\_LATE*  
 Problems with less lateness than this are excluded.  
 Default: 0

*item\_request\_filter* - - *a* Expression field of model *REQUEST\_PLANNED\_LATE*  
 An Expression that is passed an *Item\_Request* as "i" and should return "true" if the *REQUEST\_PLANNED\_LATE* Problems with that *Item\_Request* should be included in the *Problem\_Set*. If it returns "false", then *REQUEST\_PLANNED\_LATE* Problems with that *Item\_Request* will be excluded.  
 Default: true

Extensions	REQUEST_PLANNED_EARLY Extension
------------	---------------------------------

**problem\_filter** -- *a Expression field of model REQUEST\_PLANNED\_LATE*  
An Expression that is passed a Problem as '#' and should return "true" if the identified REQUEST\_PLANNED\_LATE Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
Default: true

**REQUEST\_PLANNED\_EARLY** -- *a category extension of model Problem\_Set*

A REQUEST\_PLANNED\_EARLY Problem\_Set selects  
REQUEST\_PLANNED\_EARLY Problems with certain characteristics.

This is a subset of the REQUEST and REQUEST\_PLAN Problem\_Set categories.

**min\_earliness** -- *a Time field of model REQUEST\_PLANNED\_EARLY*  
Problems with less earliness than this are excluded.  
Default: 0

**item\_request\_filter** -- *a Expression field of model REQUEST\_PLANNED\_EARLY*

An Expression that is passed an Item\_Request as '#' and should return "true" if the REQUEST\_PLANNED\_EARLY Problems with that Item\_Request should be included in the Problem\_Set. If it returns "false", then REQUEST\_PLANNED\_EARLY Problems with that Item\_Request will be excluded.  
Default: true

**problem\_filter** -- *a Expression field of model REQUEST\_PLANNED\_EARLY*  
An Expression that is passed a Problem as '#' and should return "true" if the identified REQUEST\_PLANNED\_EARLY Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
Default: true

**REQUEST\_PLANNED\_SHORT** -- *a category extension of model Problem\_Set*

A REQUEST\_PLANNED\_SHORT Problem\_Set selects  
REQUEST\_PLANNED\_SHORT Problems with certain characteristics.

This is a subset of the REQUEST and REQUEST\_PLAN Problem\_Set categories.

**min\_shortness** -- *a Quantity field of model REQUEST\_PLANNED\_SHORT*  
Problems with less shortness than this are excluded.  
Default: 0

Extensions	REQUEST_PLANNED_EXCESS Extension
------------	----------------------------------

**item\_request\_filter** -- *a Expression field of model REQUEST\_PLANNED\_SHORT*  
An Expression that is passed an Item\_Request as '#' and should return "true" if the REQUEST\_PLANNED\_SHORT Problems with that Item\_Request should be included in the Problem\_Set. If it returns "false", then REQUEST\_PLANNED\_SHORT Problems with that Item\_Request will be excluded.  
Default: true

**problem\_filter** -- *a Expression field of model REQUEST\_PLANNED\_SHORT*  
An Expression that is passed a Problem as '#' and should return "true" if the identified REQUEST\_PLANNED\_SHORT Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
Default: true

**REQUEST\_PLANNED\_EXCESS** -- *a category extension of model Problem\_Set*

A REQUEST\_PLANNED\_EXCESS Problem\_Set selects  
REQUEST\_PLANNED\_EXCESS Problems with certain characteristics.

This is a subset of the REQUEST and REQUEST\_PLAN Problem\_Set categories.

**min\_excess** -- *a Quantity field of model REQUEST\_PLANNED\_EXCESS*  
Problems with less than this amount of excess are excluded.  
Default: 0

**item\_request\_filter** -- *a Expression field of model REQUEST\_PLANNED\_EXCESS*  
An Expression that is passed an Item\_Request as '#' and should return "true" if the REQUEST\_PLANNED\_EXCESS Problems with that Item\_Request should be included in the Problem\_Set. If it returns "false", then REQUEST\_PLANNED\_EXCESS Problems with that Item\_Request will be excluded.  
Default: true

**problem\_filter** -- *a Expression field of model REQUEST\_PLANNED\_EXCESS*  
An Expression that is passed a Problem as '#' and should return "true" if the identified REQUEST\_PLANNED\_EXCESS Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
Default: true

**PROMISE\_NOT\_PLANNED** -- *a category extension of model Problem\_Set*



A PROMISE\_NOT\_PLANNED Problem\_Set selects PROMISE\_NOT\_PLANNED Problems with certain characteristics.

This is a subset of the PROMISE Problem\_Set category.

**item\_promise\_filter** - - *a Expression field of model PROMISE\_NOT\_PLANNED*  
An Expression that is passed an Item\_Promise as '#' and should return "true" if the PROMISE\_NOT\_PLANNED Problems with that Item\_Promise should be included in the Problem\_Set. If it returns "false", then PROMISE\_NOT\_PLANNED Problems with that Item\_Promise will be excluded.  
Default: true

**problem\_filter** - - *a Expression field of model PROMISE\_NOT\_PLANNED*  
An Expression that is passed a Problem as '#' and should return "true" if the identified PROMISE\_NOT\_PLANNED Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
Default: true

**PROMISE\_PLANNED\_LATE** - - *a category extension of model Problem\_Set*

A PROMISE\_PLANNED\_LATE Problem\_Set selects PROMISE\_PLANNED\_LATE Problems with certain characteristics.

This is a subset of the PROMISE and PROMISE\_PLAN Problem\_Set categories.

**min\_lateness** - - *a Time field of model PROMISE\_PLANNED\_LATE*  
Problems with less lateness than this are excluded.  
Default: 0

**item\_promise\_filter** - - *a Expression field of model PROMISE\_PLANNED\_LATE*  
An Expression that is passed an Item\_Promise as '#' and should return "true" if the PROMISE\_PLANNED\_LATE Problems with that Item\_Promise should be included in the Problem\_Set. If it returns "false", then PROMISE\_PLANNED\_LATE Problems with that Item\_Promise will be excluded.  
Default: true

**problem\_filter** - - *a Expression field of model PROMISE\_PLANNED\_LATE*  
An Expression that is passed a Problem as '#' and should return "true" if the identified PROMISE\_PLANNED\_LATE Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
Default: true

**PROMISE\_PLANNED\_EARLY** - - *a category extension of model Problem\_Set*

A PROMISE\_PLANNED\_EARLY Problem\_Set selects PROMISE\_PLANNED\_EARLY Problems with certain characteristics.

This is a subset of the PROMISE and PROMISE\_PLAN Problem\_Set categories.

**min\_earliness** - - *a Time field of model PROMISE\_PLANNED\_EARLY*  
Problems with less earliness than this are excluded.  
Default: 0

**item\_promise\_filter** - - *a Expression field of model PROMISE\_PLANNED\_EARLY*  
An Expression that is passed an Item\_Promise as '#' and should return "true" if the PROMISE\_PLANNED\_EARLY Problems with that Item\_Promise should be included in the Problem\_Set. If it returns "false", then PROMISE\_PLANNED\_EARLY Problems with that Item\_Promise will be excluded.  
Default: true

**problem\_filter** - - *a Expression field of model PROMISE\_PLANNED\_EARLY*  
An Expression that is passed a Problem as '#' and should return "true" if the identified PROMISE\_PLANNED\_EARLY Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
Default: true

**PROMISE\_PLANNED\_SHORT** - - *a category extension of model Problem\_Set*

A PROMISE\_PLANNED\_SHORT Problem\_Set selects PROMISE\_PLANNED\_SHORT Problems with certain characteristics.

This is a subset of the PROMISE and PROMISE\_PLAN Problem\_Set categories.

**min\_shortness** - - *a Quantity field of model PROMISE\_PLANNED\_SHORT*  
Problems with less shortness than this are excluded.  
Default: 0

<i>Extensions</i>	<i>PROMISE_PLANNED_EXCESS Extension</i>
-------------------	---

*Item\_promise\_filter* -- *a Expression field of model PROMISE\_PLANNED\_SHORT*  
 An Expression that is passed an Item\_Promise as '#' and should return "true" if the PROMISE\_PLANNED\_SHORT Problems with that Item\_Promise should be included in the Problem\_Set. If it returns "false", then PROMISE\_PLANNED\_SHORT Problems with that Item\_Promise will be excluded.  
 Default: true

*problem\_filter* -- *a Expression field of model PROMISE\_PLANNED\_SHORT*  
 An Expression that is passed a Problem as '#' and should return "true" if the identified PROMISE\_PLANNED\_SHORT Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default: true

**PROMISE\_PLANNED\_EXCESS** -- *a category extension of model Problem\_Set*  
 A PROMISE\_PLANNED\_EXCESS Problem\_Set selects  
 PROMISE\_PLANNED\_EXCESS Problems with certain characteristics.

This is a subset of the PROMISE and PROMISE\_PLAN Problem\_Set categories.  
*min\_excess* -- *a Quantity field of model PROMISE\_PLANNED\_EXCESS*  
 Problems with less excess than this are excluded.  
 Default: 0

*Item\_promise\_filter* -- *a Expression field of model PROMISE\_PLANNED\_EXCESS*  
 An Expression that is passed an Item\_Promise as '#' and should return "true" if the PROMISE\_PLANNED\_EXCESS Problems with that Item\_Promise should be included in the Problem\_Set. If it returns "false", then PROMISE\_PLANNED\_EXCESS Problems with that Item\_Promise will be excluded.  
 Default: true

*problem\_filter* -- *a Expression field of model PROMISE\_PLANNED\_EXCESS*  
 An Expression that is passed a Problem as '#' and should return "true" if the identified PROMISE\_PLANNED\_EXCESS Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default: true

**ACCEPTANCE\_NOT\_PLANNED** -- *a category extension of model Problem\_Set*

<i>Extensions</i>	<i>ACCEPTANCE_PLANNED_LATE Extension</i>
-------------------	--

A ACCEPTANCE\_NOT\_PLANNED Problem\_Set selects  
 ACCEPTANCE\_NOT\_PLANNED Problems with certain characteristics.

This is a subset of the ACCEPTANCE Problem\_Set category.

*Item\_acceptance\_filter* -- *a Expression field of model ACCEPTANCE\_NOT\_PLANNED*  
 An Expression that is passed an Item\_Acceptance as '#' and should return "true" if the ACCEPTANCE\_NOT\_PLANNED Problems with that Item\_Acceptance should be included in the Problem\_Set. If it returns "false", then ACCEPTANCE\_NOT\_PLANNED Problems with that Item\_Acceptance will be excluded.  
 Default: true

*problem\_filter* -- *a Expression field of model ACCEPTANCE\_NOT\_PLANNED*  
 An Expression that is passed a Problem as '#' and should return "true" if the identified ACCEPTANCE\_NOT\_PLANNED Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default: true

**ACCEPTANCE\_PLANNED\_LATE** -- *a category extension of model Problem\_Set*  
 A ACCEPTANCE\_PLANNED\_LATE Problem\_Set selects  
 ACCEPTANCE\_PLANNED\_LATE Problems with certain characteristics.

This is a subset of the ACCEPTANCE and ACCEPTANCE\_PLAN Problem\_Set categories.  
*min\_lateness* -- *a Time field of model ACCEPTANCE\_PLANNED\_LATE*  
 Problems with less lateness than this are excluded.  
 Default: 0

*Item\_acceptance\_filter* -- *a Expression field of model ACCEPTANCE\_PLANNED\_LATE*  
 An Expression that is passed an Item\_Acceptance as '#' and should return "true" if the ACCEPTANCE\_PLANNED\_LATE Problems with that Item\_Acceptance should be included in the Problem\_Set. If it returns "false", then ACCEPTANCE\_PLANNED\_LATE Problems with that Item\_Acceptance will be excluded.  
 Default: true

<i>Extensions</i>	<i>ACCEPTANCE_PLANNED_EARLY Extension</i>
-------------------	---

**problem\_filter** - - *a Expression field of model* **ACCEPTANCE\_PLANNED\_LATE**  
 An Expression that is passed a Problem as '#' and should return "true" if the identified **ACCEPTANCE\_PLANNED\_LATE** Problems should be included in the Problem\_Set.  
 If it returns "false", then the identified Problems will be excluded.  
 Default: true

**ACCEPTANCE\_PLANNED\_EARLY** - - *a category extension of model* **Problem\_Set**  
 A **ACCEPTANCE\_PLANNED\_EARLY** Problem\_Set selects **ACCEPTANCE\_PLANNED\_EARLY** Problems with certain characteristics.

This is a subset of the **ACCEPTANCE** and **ACCEPTANCE\_PLAN** Problem\_Set categories.

**min\_earliness** - - *a Time field of model* **ACCEPTANCE\_PLANNED\_EARLY**  
 Problems with less earliness than this are excluded.  
 Default: 0

**item\_acceptance\_filter** - - *a Expression field of model*  
**ACCEPTANCE\_PLANNED\_EARLY**  
 An Expression that is passed an Item\_Acceptance as '#' and should return "true" if the **ACCEPTANCE\_PLANNED\_EARLY** Problems with that Item\_Acceptance should be included in the Problem\_Set. If it returns "false", then **ACCEPTANCE\_PLANNED\_EARLY** Problems with that Item\_Acceptance will be excluded.  
 Default: true

**problem\_filter** - - *a Expression field of model*  
**ACCEPTANCE\_PLANNED\_EARLY**  
 An Expression that is passed a Problem as '#' and should return "true" if the identified **ACCEPTANCE\_PLANNED\_EARLY** Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default: true

**ACCEPTANCE\_PLANNED\_SHORT** - - *a category extension of model* **Problem\_Set**  
 A **ACCEPTANCE\_PLANNED\_SHORT** Problem\_Set selects **ACCEPTANCE\_PLANNED\_SHORT** Problems with certain characteristics.  
 This is a subset of the **ACCEPTANCE** and **ACCEPTANCE\_PLAN** Problem\_Set categories.

<i>Extensions</i>	<i>ACCEPTANCE_PLANNED_EXCESS Extension</i>
-------------------	--

**min\_shortness** - - *a Quantity field of model* **ACCEPTANCE\_PLANNED\_SHORT**  
 Problems with less shortness than this are excluded.  
 Default: 0

**item\_acceptance\_filter** - - *a Expression field of model*  
**ACCEPTANCE\_PLANNED\_SHORT**  
 An Expression that is passed an Item\_Acceptance as '#' and should return "true" if the **ACCEPTANCE\_PLANNED\_SHORT** Problems with that Item\_Acceptance should be included in the Problem\_Set. If it returns "false", then **ACCEPTANCE\_PLANNED\_SHORT** Problems with that Item\_Acceptance will be excluded.  
 Default: true

**problem\_filter** - - *a Expression field of model*  
**ACCEPTANCE\_PLANNED\_SHORT**  
 An Expression that is passed a Problem as '#' and should return "true" if the identified **ACCEPTANCE\_PLANNED\_SHORT** Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default: true

**ACCEPTANCE\_PLANNED\_EXCESS** - - *a category extension of model* **Problem\_Set**  
 A **ACCEPTANCE\_PLANNED\_EXCESS** Problem\_Set selects **ACCEPTANCE\_PLANNED\_EXCESS** Problems with certain characteristics.

This is a subset of the **ACCEPTANCE** and **ACCEPTANCE\_PLAN** Problem\_Set categories.

**min\_excess** - - *a Quantity field of model* **ACCEPTANCE\_PLANNED\_EXCESS**  
 Problems with less than this amount of excess are excluded.  
 Default: 0

**item\_acceptance\_filter** - - *a Expression field of model*  
**ACCEPTANCE\_PLANNED\_EXCESS**  
 An Expression that is passed an Item\_Acceptance as '#' and should return "true" if the **ACCEPTANCE\_PLANNED\_EXCESS** Problems with that Item\_Acceptance should be included in the Problem\_Set. If it returns "false", then **ACCEPTANCE\_PLANNED\_EXCESS** Problems with that Item\_Acceptance will be excluded.  
 Default: true

<i>Extensions</i>	<i>PLANNED_BEFORE_CURRENT Extension</i>
-------------------	---

*problem\_filter* - - *a Expression field of model*  
**ACCEPTANCE\_PLANNED\_EXCESS**  
 An Expression that is passed a Problem as '#' and should return "true" if the identified **ACCEPTANCE\_PLANNED\_EXCESS** Problems should be included in the *Problem\_Set*. If it returns "false", then the indentified Problems will be excluded.  
 Default: true

**PLANNED\_BEFORE\_CURRENT** -- *a category extension of model Problem\_Set*

A **PLANNED\_BEFORE\_CURRENT** Problem\_Set selects **PLANNED\_BEFORE\_CURRENT** Problems on certain Operations.

*operation\_plan\_filter* - - *a Expression field of model*  
**PLANNED\_BEFORE\_CURRENT**  
 An Expression that is passed an *Operation\_Plan* as '#' and should return "true" if the Problems on that Operation should be included in the *Problem\_Set*. If it returns "false", then Problems on that Operation will be excluded.  
 Default: true

*problem\_filter* - - *a Expression field of model* **PLANNED\_BEFORE\_CURRENT**  
 An Expression that is passed a Problem as '#' and should return "true" if the identified **PLANNED\_BEFORE\_CURRENT** Problems should be included in the *Problem\_Set*. If it returns "false", then the indentified Problems will be excluded.  
 Default: true

**UNRELEASED** -- *a category extension of model Problem\_Set*

An **UNRELEASED** *Problem\_Set* selects **UNRELEASED** Problems on certain Operations.

*operation\_plan\_filter* - - *a Expression field of model* **UNRELEASED**  
 An Expression that is passed an *Operation\_Plan* as '#' and should return "true" if the Problems on that Operation should be included in the *Problem\_Set*. If it returns "false", then Problems on that Operation will be excluded.  
 Default: true

*problem\_filter* - - *a Expression field of model* **UNRELEASED**  
 An Expression that is passed a Problem as '#' and should return "true" if the identified **UNRELEASED** Problems should be included in the *Problem\_Set*. If it returns "false", then the indentified Problems will be excluded.  
 Default: true

<i>Extensions</i>	<i>NEEDS_RELEASE Extension</i>
-------------------	--------------------------------

**NEEDS\_RELEASE** -- *a category extension of model Problem\_Set*

A **NEEDS\_RELEASE** *Problem\_Set* selects **NEEDS\_RELEASE** Problems on certain Operations.

*operation\_plan\_filter* - - *a Expression field of model* **NEEDS\_RELEASE**  
 An Expression that is passed an *Operation\_Plan* as '#' and should return "true" if the Problems on that Operation should be included in the *Problem\_Set*. If it returns "false", then Problems on that Operation will be excluded.  
 Default: true

*problem\_filter* - - *a Expression field of model* **NEEDS\_RELEASE**  
 An Expression that is passed a Problem as '#' and should return "true" if the identified **NEEDS\_RELEASE** Problems should be included in the *Problem\_Set*. If it returns "false", then the indentified Problems will be excluded.  
 Default: true

**INCONSISTENT\_OPPLAN** -- *a category extension of model Problem\_Set*

A **INCONSISTENT\_OPPLAN** *Problem\_Set* selects **INCONSISTENT\_OPPLAN** Problems on certain Operations.

*operation\_plan\_filter* - - *a Expression field of model* **INCONSISTENT\_OPPLAN**  
 An Expression that is passed an *Operation\_Plan* as '#' and should return "true" if the Problems on that Operation should be included in the *Problem\_Set*. If it returns "false", then Problems on that Operation will be excluded.  
 Default: true

*problem\_filter* - - *a Expression field of model* **INCONSISTENT\_OPPLAN**  
 An Expression that is passed a Problem as '#' and should return "true" if the identified **INCONSISTENT\_OPPLAN** Problems should be included in the *Problem\_Set*. If it returns "false", then the indentified Problems will be excluded.  
 Default: true

**OPERATION** -- *a category extension of model Problem\_Set*

A **OPERATION** *Problem\_Set* selects **OPERATION** Problems on certain Operations.

operation\_plan\_filter - - *a Expression field of model OPERATION*

An Expression that is passed an Operation\_Plan as '#' and should return "true" if the Problems on that Operation should be included in the Problem\_Set. If it returns "false", then Problems on that Buffer will be excluded.  
Default: true

problem\_filter - - *a Expression field of model OPERATION*

An Expression that is passed a Problem as '#' and should return "true" if the identified OPERATION Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
Default: true

REQUEST\_PROMISED\_LATE - - *a category extension of model Problem\_Set*

A REQUEST\_PROMISED\_LATE Problem\_Set selects REQUEST\_PROMISED\_LATE Problems with certain characteristics.

This is a subset of the REQUEST\_PROMISE, and REQUEST\_PROMISE Problem\_Set categories.

min\_latency - - *a Time field of model REQUEST\_PROMISED\_LATE*  
Problems with less latency than this are excluded.

Default: 0

delivery\_promise\_filter - - *a Expression field of model REQUEST\_PROMISED\_LATE*

An Expression that is passed a Delivery\_Promise as '#' and should return "true" if the REQUEST\_PROMISED\_LATE Problems with that Delivery\_Promise should be included in the Problem\_Set. If it returns "false", then REQUEST\_PROMISED\_LATE Problems with that Delivery\_Promise will be excluded.  
Default: true

problem\_filter - - *a Expression field of model REQUEST\_PROMISED\_LATE*

An Expression that is passed a Problem as '#' and should return "true" if the identified REQUEST\_PROMISED\_LATE Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
Default: true

REQUEST\_PROMISED\_EARLY - - *a category extension of model Problem\_Set*

A REQUEST\_PROMISED\_EARLY Problem\_Set selects REQUEST\_PROMISED\_EARLY Problems with certain characteristics.

This is a subset of the REQUEST\_PROMISE, and REQUEST\_PROMISE Problem\_Set categories.

min\_earliness - - *a Time field of model REQUEST\_PROMISED\_EARLY*  
Problems with less earliness than this are excluded.

Default: 0

delivery\_promise\_filter - - *a Expression field of model REQUEST\_PROMISED\_EARLY*

An Expression that is passed a Delivery\_Promise as '#' and should return "true" if the REQUEST\_PROMISED\_EARLY Problems with that Delivery\_Promise should be included in the Problem\_Set. If it returns "false", then REQUEST\_PROMISED\_EARLY Problems with that Delivery\_Promise will be excluded.  
Default: true

problem\_filter - - *a Expression field of model REQUEST\_PROMISED\_EARLY*  
An Expression that is passed a Problem as '#' and should return "true" if the identified REQUEST\_PROMISED\_EARLY Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
Default: true

REQUEST\_PROMISED\_SHORT - - *a category extension of model Problem\_Set*

A REQUEST\_PROMISED\_SHORT Problem\_Set selects REQUEST\_PROMISED\_SHORT Problems with certain characteristics.

This is a subset of the REQUEST\_PROMISE, and REQUEST\_PROMISE Problem\_Set categories.

min\_shortness - - *a Quantity field of model REQUEST\_PROMISED\_SHORT*  
Problems with less shortness than this are excluded.

Default: 0

<i>Extensions</i>	<i>REQUEST_PROMISED_EXCESS Extension</i>
-------------------	--

**Item\_promise\_filter** -- *a Expression field of model REQUEST\_PROMISED\_SHORT*  
 An Expression that is passed an Item\_Promise as '#' and should return "true" if the REQUEST\_PROMISED\_SHORT Problems with that Item\_Promise should be included in the Problem\_Set. If it returns "false", then REQUEST\_PROMISED\_SHORT Problems with that Item\_Promise will be excluded.  
 Default: true

**problem\_filter** -- *a Expression field of model REQUEST\_PROMISED\_SHORT*  
 An Expression that is passed a Problem as '#' and should return "true" if the identified REQUEST\_PROMISED\_SHORT Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default: true

**REQUEST\_PROMISED\_EXCESS** -- *a category extension of model Problem\_Set*  
 A REQUEST\_PROMISED\_EXCESS Problem\_Set selects REQUEST\_PROMISED\_EXCESS Problems with certain characteristics.

This is a subset of the REQUEST, PROMISE, and REQUEST\_PROMISE Problem\_Set categories.

**min\_excess** -- *a Quantity field of model REQUEST\_PROMISED\_EXCESS*  
 Problems with less excess than this are excluded.  
 Default: 0

**Item\_promise\_filter** -- *a Expression field of model REQUEST\_PROMISED\_EXCESS*  
 An Expression that is passed an Item\_Promise as '#' and should return "true" if the REQUEST\_PROMISED\_EXCESS Problems with that Item\_Promise should be included in the Problem\_Set. If it returns "false", then REQUEST\_PROMISED\_EXCESS Problems with that Item\_Promise will be excluded.  
 Default: true

**problem\_filter** -- *a Expression field of model REQUEST\_PROMISED\_EXCESS*  
 An Expression that is passed a Problem as '#' and should return "true" if the identified REQUEST\_PROMISED\_EXCESS Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default: true

<i>Extensions</i>	<i>PROMISE_NOT_OFFERED Extension</i>
-------------------	--------------------------------------

**PROMISE\_NOT\_OFFERED** -- *a category extension of model Problem\_Set*

A PROMISE\_NOT\_OFFERED Problem\_Set selects PROMISE\_NOT\_OFFERED Problems with certain characteristics.

This is a subset of the REQUEST and PROMISE Problem\_Set categories.

**request\_filter** -- *a Expression field of model PROMISE\_NOT\_OFFERED*  
 An Expression that is passed a Request as '#' and should return "true" if the Problems with that Request should be included in the Problem\_Set. If it returns "false", then Problems with that Request will be excluded.  
 Default: true

**problem\_filter** -- *a Expression field of model PROMISE\_NOT\_OFFERED*  
 An Expression that is passed a Problem as '#' and should return "true" if the identified PROMISE\_NOT\_OFFERED Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default: true

**PROMISE\_NOT\_ACCEPTED** -- *a category extension of model Problem\_Set*

A PROMISE\_NOT\_ACCEPTED Problem\_Set selects PROMISE\_NOT\_ACCEPTED Problems with certain characteristics.

This is a subset of the REQUEST and PROMISE Problem\_Set categories.

**request\_filter** -- *a Expression field of model PROMISE\_NOT\_ACCEPTED*  
 An Expression that is passed a Request as '#' and should return "true" if the Problems with that Request should be included in the Problem\_Set. If it returns "false", then Problems with that Request will be excluded.  
 Default: true

**problem\_filter** -- *a Expression field of model PROMISE\_NOT\_ACCEPTED*  
 An Expression that is passed a Problem as '#' and should return "true" if the identified PROMISE\_NOT\_ACCEPTED Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default: true

**ACCEPTANCE\_INCONSISTENT** -- *a category extension of model Problem\_Set*

Extensions	REQUEST_QUEUED Extension
------------	--------------------------

A ACCEPTANCE\_INCONSISTENT Problem\_Set selects ACCEPTANCE\_INCONSISTENT Problems with certain characteristics.

This is a subset of the REQUEST and PROMISE Problem\_Set categories.

**request\_filter** -- *a Expression field of model ACCEPTANCE\_INCONSISTENT*  
An Expression that is passed a Request as '#' and should return "true" if the Problems with that Request should be included in the Problem\_Set. If it returns "false", then Problems with that Request will be excluded.  
Default: true

**problem\_filter** -- *a Expression field of model ACCEPTANCE\_INCONSISTENT*  
An Expression that is passed a Problem as '#' and should return "true" if the identified ACCEPTANCE\_INCONSISTENT Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
Default: true

**REQUEST\_QUEUED** -- *a category extension of model Problem\_Set*

A REQUEST\_QUEUED Problem\_Set selects REQUEST\_QUEUED Problems with certain characteristics.

This is a subset of the REQUEST Problem\_Set category.

**request\_filter** -- *a Expression field of model REQUEST\_QUEUED*  
An Expression that is passed a Request as '#' and should return "true" if the Problems with that Request should be included in the Problem\_Set. If it returns "false", then Problems with that Request will be excluded.  
Default: true

**problem\_filter** -- *a Expression field of model REQUEST\_QUEUED*  
An Expression that is passed a Problem as '#' and should return "true" if the identified REQUEST\_QUEUED Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
Default: true

**REQUEST** -- *a category extension of model Problem\_Set*

A REQUEST Problem\_Set selects Request Problems (e.g., REQUEST\_PLANNED\_LATE, etc.) on certain REQUESTS.

Extensions	REQUEST_PLAN Extension
------------	------------------------

**request\_filter** -- *a Expression field of model REQUEST*  
An Expression that is passed a Request as '#' and should return "true" if the Problems with that Request should be included in the Problem\_Set. If it returns "false", then Problems with that Request will be excluded.  
Default: true

**problem\_filter** -- *a Expression field of model REQUEST*  
An Expression that is passed a Problem as '#' and should return "true" if the identified REQUEST Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
Default: true

**REQUEST\_PLAN** -- *a category extension of model Problem\_Set*

A REQUEST\_PLAN Problem\_Set selects Problems between Item\_Requests and their delivery plan, including REQUEST\_PLANNED\_LATE, REQUEST\_PLANNED\_EARLY, REQUEST\_PLANNED\_SHORT, and REQUEST\_PLANNED\_EXCESS.

This is a subset of the REQUEST Problem\_Set category.

**item\_request\_filter** -- *a Expression field of model REQUEST\_PLAN*  
An Expression that is passed a Item\_Request as '#' and should return "true" if the Problems with that Item\_Request should be included in the Problem\_Set. If it returns "false", then Problems with that Item\_Request will be excluded.  
Default: true

**problem\_filter** -- *a Expression field of model REQUEST\_PLAN*  
An Expression that is passed a Problem as '#' and should return "true" if the identified REQUEST\_PLAN Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
Default: true

**PROMISE** -- *a category extension of model Problem\_Set*

A PROMISE Problem\_Set selects Promise Problems.

**promise\_filter** -- *a Expression field of model PROMISE*  
An Expression that is passed a Promise as '#' and should return "true" if the Problems with that Promise should be included in the Problem\_Set. If it returns "false", then Problems with that Promise will be excluded.  
Default: true

**problem\_filter** -- *a Expression field of model PROMISE*

An Expression that is passed a Problem as '#' and should return "true" if the identified PROMISE Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

**PROMISE\_PLAN** -- *a category extension of model Problem\_Set*

A PROMISE\_PLAN Problem\_Set selects Problems between Item\_Promises and their delivery plan, including PROMISE\_PLANNED\_LATE, PROMISE\_PLANNED\_EARLY, PROMISE\_PLANNED\_SHORT, and PROMISE\_PLANNED\_EXCESS.

This is a subset of the PROMISE Problem\_Set category.

**Item\_Promise\_Filter** -- *a Expression field of model PROMISE\_PLAN*

An Expression that is passed a Item\_Promise as '#' and should return "true" if the Problems with that Item\_Promise should be included in the Problem\_Set. If it returns "false", then Problems with that Item\_Promise will be excluded.

Default: true

**problem\_filter** -- *a Expression field of model PROMISE\_PLAN*

An Expression that is passed a Problem as '#' and should return "true" if the identified PROMISE\_PLAN Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

**REQUEST\_PROMISE** -- *a category extension of model Problem\_Set*

A REQUEST\_PROMISE\_PLAN Problem\_Set selects Problems that result from discrepancies between a Request and the Promise offered to it, including REQUEST\_PROMISED\_LATE, REQUEST\_PROMISED\_EARLY, REQUEST\_PROMISED\_SHORT, REQUEST\_PROMISED\_EXCESS, DELIVERY\_REQUEST\_OVERPRICED, and ITEM\_REQUEST\_OVERPRICED.

This is a subset of both the REQUEST and PROMISE Problem\_Set categories.

**delivery\_promise\_filter** -- *a Expression field of model REQUEST\_PROMISE*

An Expression that is passed a Delivery\_Promise as '#' and should return "true" if the Problems with that Delivery\_Promise should be included in the Problem\_Set. If it returns "false", then Problems with that Delivery\_Promise will be excluded.

Default: true

**problem\_filter** -- *a Expression field of model REQUEST\_PROMISE*

An Expression that is passed a Problem as '#' and should return "true" if the identified REQUEST\_PROMISE Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

**DELIVERY\_REQUEST\_NOT\_COORDINATED** -- *a category extension of model Problem\_Set*

A DELIVERY\_REQUEST\_NOT\_COORDINATED Problem\_Set selects Problems that result from items of a Delivery\_Request not all being scheduled for delivery on the same date.

**problem\_filter** -- *a Expression field of model***DELIVERY\_REQUEST\_NOT\_COORDINATED**

An Expression that is passed a Problem as '#' and should return "true" if the identified DELIVERY\_REQUEST\_NOT\_COORDINATED Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

**DELIVERY\_PROMISE\_NOT\_COORDINATED** -- *a category extension of model Problem\_Set*

A DELIVERY\_PROMISE\_NOT\_COORDINATED Problem\_Set selects Problems that result from items of a Delivery\_Promise not all being scheduled for delivery on the same date.

**problem\_filter** -- *a Expression field of model***DELIVERY\_PROMISE\_NOT\_COORDINATED**

An Expression that is passed a Problem as '#' and should return "true" if the identified DELIVERY\_PROMISE\_NOT\_COORDINATED Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

**DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED** -- *a category extension of model Problem\_Set*

A DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED Problem\_Set selects Problems that result from items of a Delivery\_Acceptance not all being scheduled for delivery on the same date.



Extensions	SUPPLY_PLANNED_LATE Extension
------------	-------------------------------

**problem\_filter** - - *a Expression field of model DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED*  
 An Expression that is passed a Problem as '#' and should return "true" if the identified DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default: true

**SUPPLY\_PLANNED\_LATE** - - *a category extension of model Problem\_Set*

A SUPPLY\_PLANNED\_LATE Problem\_Set selects SUPPLY\_PLANNED\_LATE Problems with certain characteristics.

This is a subset of the SUPPLY and SUPPLY\_PLAN Problem\_Set categories.

**min\_lateness** - - *a Time field of model SUPPLY\_PLANNED\_LATE*  
 Problems with less lateness than this are excluded.  
 Default: 0

**item\_request\_filter** - - *a Expression field of model SUPPLY\_PLANNED\_LATE*  
 An Expression that is passed an Item\_Request as '#' and should return "true" if the SUPPLY\_PLANNED\_LATE Problems with that Item\_Request should be included in the Problem\_Set. If it returns "false", then SUPPLY\_PLANNED\_LATE Problems with that Item\_Request will be excluded.  
 Default: true

**problem\_filter** - - *a Expression field of model SUPPLY\_PLANNED\_LATE*  
 An Expression that is passed a Problem as '#' and should return "true" if the identified SUPPLY\_PLANNED\_LATE Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default: true

**SUPPLY\_PLANNED\_EARLY** - - *a category extension of model Problem\_Set*

A SUPPLY\_PLANNED\_EARLY Problem\_Set selects SUPPLY\_PLANNED\_EARLY Problems with certain characteristics.

This is a subset of the SUPPLY and SUPPLY\_PLAN Problem\_Set categories.  
**min\_cairliness** - - *a Time field of model SUPPLY\_PLANNED\_EARLY*  
 Problems with less cairliness than this are excluded.  
 Default: 0

Extensions	SUPPLY_PLANNED_SHORT Extension
------------	--------------------------------

**item\_request\_filter** - - *a Expression field of model SUPPLY\_PLANNED\_EARLY*  
 An Expression that is passed an Item\_Request as '#' and should return "true" if the SUPPLY\_PLANNED\_EARLY Problems with that Item\_Request should be included in the Problem\_Set. If it returns "false", then SUPPLY\_PLANNED\_EARLY Problems with that Item\_Request will be excluded.  
 Default: true

**problem\_filter** - - *a Expression field of model SUPPLY\_PLANNED\_EARLY*

An Expression that is passed a Problem as '#' and should return "true" if the identified SUPPLY\_PLANNED\_EARLY Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default: true

**SUPPLY\_PLANNED\_SHORT** - - *a category extension of model Problem\_Set*

A SUPPLY\_PLANNED\_SHORT Problem\_Set selects SUPPLY\_PLANNED\_SHORT Problems with certain characteristics.

This is a subset of the SUPPLY and SUPPLY\_PLAN Problem\_Set categories.

**min\_shortness** - - *a Quantity field of model SUPPLY\_PLANNED\_SHORT*  
 Problems with less shortness than this are excluded.  
 Default: 0

**item\_request\_filter** - - *a Expression field of model SUPPLY\_PLANNED\_SHORT*  
 An Expression that is passed an Item\_Request as '#' and should return "true" if the SUPPLY\_PLANNED\_SHORT Problems with that Item\_Request should be included in the Problem\_Set. If it returns "false", then SUPPLY\_PLANNED\_SHORT Problems with that Item\_Request will be excluded.  
 Default: true

**problem\_filter** - - *a Expression field of model SUPPLY\_PLANNED\_SHORT*  
 An Expression that is passed a Problem as '#' and should return "true" if the identified SUPPLY\_PLANNED\_SHORT Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default: true

**SUPPLY\_PLANNED\_EXCESS** - - *a category extension of model Problem\_Set*

A SUPPLY\_PLANNED\_EXCESS Problem\_Set selects SUPPLY\_PLANNED\_EXCESS Problems with certain characteristics.

This is a subset of the SUPPLY and SUPPLY\_PLAN Problem\_Set categories.

**min\_excess** - - *a Quantity field of model SUPPLY\_PLANNED\_EXCESS*  
Problems with less than this amount of excess are excluded.

Default: 0

**item\_request\_filter** - - *a Expression field of model SUPPLY\_PLANNED\_EXCESS*  
An Expression that is passed an Item\_Request as '#' and should return "true" if the SUPPLY\_PLANNED\_EXCESS Problems with that Item\_Request should be included in the Problem\_Set. If it returns "false", then SUPPLY\_PLANNED\_EXCESS Problems with that Item\_Request will be excluded.

Default: true

**problem\_filter** - - *a Expression field of model SUPPLY\_PLANNED\_EXCESS*  
An Expression that is passed a Problem as '#' and should return "true" if the identified SUPPLY\_PLANNED\_EXCESS Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

**SUPPLY\_PROMISED\_LATE** - - *a category extension of model Problem\_Set*

A SUPPLY\_PROMISED\_LATE Problem\_Set selects SUPPLY\_PROMISED\_LATE Problems with certain characteristics.

This is a subset of the SUPPLY and SUPPLY\_PROMISE Problem\_Set categories.

**min\_lateness** - - *a Time field of model SUPPLY\_PROMISED\_LATE*  
Problems with less lateness than this are excluded.

Default: 0

**delivery\_promise\_filter** - - *a Expression field of model SUPPLY\_PROMISED\_LATE*

An Expression that is passed an Delivery\_Promise as '#' and should return "true" if the SUPPLY\_PROMISED\_LATE Problems with that Delivery\_Promise should be included in the Problem\_Set. If it returns "false", then SUPPLY\_PROMISED\_LATE Problems with that Delivery\_Promise will be excluded.

Default: true

**problem\_filter** - - *a Expression field of model SUPPLY\_PROMISED\_LATE*  
An Expression that is passed a Problem as '#' and should return "true" if the identified SUPPLY\_PROMISED\_LATE Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

**SUPPLY\_PROMISED\_EARLY** - - *a category extension of model Problem\_Set*

A SUPPLY\_PROMISED\_EARLY Problem\_Set selects SUPPLY\_PROMISED\_EARLY Problems with certain characteristics.

This is a subset of the SUPPLY and SUPPLY\_PROMISE Problem\_Set categories.

**min\_earliness** - - *a Time field of model SUPPLY\_PROMISED\_EARLY*  
Problems with less earliness than this are excluded.

Default: 0

**delivery\_promise\_filter** - - *a Expression field of model SUPPLY\_PROMISED\_EARLY*

An Expression that is passed an Delivery\_Promise as '#' and should return "true" if the SUPPLY\_PROMISED\_EARLY Problems with that Delivery\_Promise should be included in the Problem\_Set. If it returns "false", then SUPPLY\_PROMISED\_EARLY Problems with that Delivery\_Promise will be excluded.

Default: true

**problem\_filter** - - *a Expression field of model SUPPLY\_PROMISED\_EARLY*  
An Expression that is passed a Problem as '#' and should return "true" if the identified SUPPLY\_PROMISED\_EARLY Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

**SUPPLY\_PROMISED\_SHORT** - - *a category extension of model Problem\_Set*

A SUPPLY\_PROMISED\_SHORT Problem\_Set selects SUPPLY\_PROMISED\_SHORT Problems with certain characteristics.

This is a subset of the SUPPLY and SUPPLY\_PROMISE Problem\_Set categories.

**min\_shortness** - - *a Quantity field of model SUPPLY\_PROMISED\_SHORT*  
Problems with less shortness than this are excluded.

Default: 0

Extensions	SUPPLY_PROMISED_EXCESS Extension
------------	----------------------------------

Item\_Promise\_Filter -- a Expression field of model  
SUPPLY\_PROMISED\_SHORT

An Expression that is passed an Item\_Promise as '#' and should return "true" if the SUPPLY\_PROMISED\_SHORT Problems with that Item\_Promise should be included in the Problem\_Set. If it returns "false", then SUPPLY\_PROMISED\_SHORT Problems with that Item\_Promise will be excluded.

Default: true

problem\_filter -- a Expression field of model SUPPLY\_PROMISED\_SHORT

An Expression that is passed a Problem as '#' and should return "true" if the identified SUPPLY\_PROMISED\_SHORT Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

SUPPLY\_PROMISED\_EXCESS -- a category extension of model Problem\_Set

A SUPPLY\_PROMISED\_EXCESS Problem\_Set selects  
SUPPLY\_PROMISED\_EXCESS Problems with certain characteristics.

This is a subset of the SUPPLY and SUPPLY\_PROMISE Problem\_Set categories.

min\_excess -- a Quantity field of model SUPPLY\_PROMISED\_EXCESS  
Problems with less excess than this are excluded.  
Default: 0

Item\_Promise\_Filter -- a Expression field of model  
SUPPLY\_PROMISED\_EXCESS

An Expression that is passed an Item\_Promise as '#' and should return "true" if the SUPPLY\_PROMISED\_EXCESS Problems with that Item\_Promise should be included in the Problem\_Set. If it returns "false", then SUPPLY\_PROMISED\_EXCESS Problems with that Item\_Promise will be excluded.

Default: true

problem\_filter -- a Expression field of model SUPPLY\_PROMISED\_EXCESS

An Expression that is passed a Problem as '#' and should return "true" if the identified SUPPLY\_PROMISED\_EXCESS Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

SUPPLY -- a category extension of model Problem\_Set

Extensions	SUPPLY_PLAN Extension
------------	-----------------------

A SUPPLY Problem\_Set selects Supply Request Problems (e.g.,  
SUPPLY\_PLANNED\_LATE, etc.) on certain supply\_requests.

request\_filter -- a Expression field of model SUPPLY

An Expression that is passed a Request as '#' and should return "true" if the Problems with that Request should be included in the Problem\_Set. If it returns "false", then Problems with that Request will be excluded.

Default: true

problem\_filter -- a Expression field of model SUPPLY

An Expression that is passed a Problem as '#' and should return "true" if the identified SUPPLY Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

SUPPLY\_PLAN -- a category extension of model Problem\_Set

A SUPPLY\_PLAN Problem\_Set selects Problems between the receiving\_plan and the delivery\_plan of supply Requests, including SUPPLY\_PLANNED\_LATE,  
SUPPLY\_PLANNED\_EARLY, SUPPLY\_PLANNED\_SHORT, and  
SUPPLY\_PLANNED\_EXCESS.

This is a subset of the SUPPLY Problem\_Set category.

Item\_Request\_Filter -- a Expression field of model SUPPLY\_PLAN

An Expression that is passed a Item\_Request as '#' and should return "true" if the Problems with that Item\_Request should be included in the Problem\_Set. If it returns "false", then Problems with that Item\_Request will be excluded.

Default: true

problem\_filter -- a Expression field of model SUPPLY\_PLAN

An Expression that is passed a Problem as '#' and should return "true" if the identified SUPPLY\_PLAN Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

SUPPLY\_PROMISE -- a category extension of model Problem\_Set

A SUPPLY\_PROMISE\_PLAN Problem\_Set selects Problems that result from discrepancies between the receiving\_plan of a supply Request's and the Promise offered to it, including SUPPLY\_PROMISED\_LATE, SUPPLY\_PROMISED\_EARLY,  
SUPPLY\_PROMISED\_SHORT, and SUPPLY\_PROMISED\_EXCESS.

This is a subset of the SUPPLY Problem\_Set category.

**delivery\_promise\_filter** -- a Expression field of model SUPPLY\_PROMISE

An Expression that is passed a Delivery\_Promise as '#' and should return "true" if the Problems with that Delivery\_Promise should be included in the Problem\_Set. If it returns "false", then Problems with that Delivery\_Promise will be excluded.

Default: true

**problem\_filter** -- a Expression field of model SUPPLY\_PROMISE

An Expression that is passed a Problem as '#' and should return "true" if the identified SUPPLY\_PROMISE Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

**UNIDENTIFIED\_OP\_STATE** -- a category extension of model Problem\_Set

A UNIDENTIFIED\_OP\_STATE Problem\_Set selects UNIDENTIFIED\_OP\_STATE Problems on certain Operations.

**operation\_plan\_filter** -- a Expression field of model UNIDENTIFIED\_OP\_STATE

An Expression that is passed an Operation\_Plan as '#' and should return "true" if the Problems on that Operation should be included in the Problem\_Set. If it returns "false", then Problems on that Operation will be excluded.

Default: true

**problem\_filter** -- a Expression field of model UNIDENTIFIED\_OP\_STATE

An Expression that is passed a Problem as '#' and should return "true" if the identified UNIDENTIFIED\_OP\_STATE Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

**UNCONSOLIDATED** -- a category extension of model Problem\_Set

Unconsolidated Operation\_Plans exist in this resource\_plan.

**resource\_plan\_filter** -- a Expression field of model UNCONSOLIDATED

An Expression that is passed a Resource\_Plan as '#' and should return "true" if the UNCONSOLIDATED Problems on that Resource should be included in the Problem\_Set. If it returns "false", then Problems on that Resource will be excluded.

Default: true

**UNCOORDINATED** -- a category extension of model Problem\_Set

Uncoordinated Operation\_Plans exist in this resource\_plan.

**resource\_plan\_filter** -- a Expression field of model UNCOORDINATED

An Expression that is passed a Resource\_Plan as '#' and should return "true" if the UNCOORDINATED Problems on that Resource should be included in the Problem\_Set. If it returns "false", then Problems on that Resource will be excluded.

Default: true

**CONSOLIDATION\_OVERSIZE** -- a category extension of model Problem\_Set

CONSOLIDATION\_OVERSIZED Operation\_Plans exist in this resource\_plan.

**resource\_plan\_filter** -- a Expression field of model CONSOLIDATION\_OVERSIZE

An Expression that is passed a Resource\_Plan as '#' and should return "true" if the CONSOLIDATION\_OVERSIZED Problems on that Resource should be included in the Problem\_Set. If it returns "false", then Problems on that Resource will be excluded.

Default: true

**CONSOLIDATION\_UNDERSIZE** -- a category extension of model Problem\_Set

CONSOLIDATION\_UNDERSIZED Operation\_Plans exist in this resource\_plan.

**resource\_plan\_filter** -- a Expression field of model CONSOLIDATION\_UNDERSIZE

An Expression that is passed a Resource\_Plan as '#' and should return "true" if the CONSOLIDATION\_UNDERSIZED Problems on that Resource should be included in the Problem\_Set. If it returns "false", then Problems on that Resource will be excluded.

Default: true

**OVERLOAD** -- a category extension of model Problem\_Set

An OVERLOAD Problem\_Set selects OVERLOAD Problems with certain characteristics.

**resource\_plan\_filter** -- a Expression field of model OVERLOAD

An Expression that is passed a Resource\_Plan as '#' and should return "true" if the OVERLOAD Problems on that Resource should be included in the Problem\_Set.

<i>Extensions</i>	<i>OVERSIZE Extension</i>
-------------------	---------------------------

Default: true

**problem\_filter** -- *a Expression field of model OVERLOAD*

An Expression that is passed a Problem as '#' and should return "true" if the identified OVERLAP Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

**min\_overload\_time** -- *a Time field of model OVERLOAD*

A Problem will be excluded if its 'overload\_time' is less than the minimum Time specified by this Time\_Horizon for the 'dates.start' of the Problem. The default, 0, will not exclude any OVERLOAD Problem.

Default: 0

**min\_overload\_std\_time** -- *a Time field of model OVERLOAD*

A Problem will be excluded if its 'overload\_std\_time' is less than the minimum Time specified by this Time\_Horizon for the 'dates.start' of the Problem. The default, 0, will not exclude any OVERLOAD Problem.

Default: 0

**OVERSIZE** -- *a category extension of model Problem\_Set*

An OVERSIZE Problem\_Set selects OVERSIZE Problems with certain characteristics.

**resource\_plan\_filter** -- *a Expression field of model OVERSIZE*

An Expression that is passed a Resource\_Plan as '#' and should return "true" if the OVERSIZE Problems on that Resource should be included in the Problem\_Set. If it returns "false", then Problems on that Resource will be excluded.

Default: true

**problem\_filter** -- *a Expression field of model OVERSIZE*

An Expression that is passed a Problem as '#' and should return "true" if the identified OVERSIZE Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.

Default: true

**min\_oversize** -- *a Percentage field of model OVERSIZE*

A Problem will be excluded if its 'oversize' is less than the minimum Percentage specified by this Percentage\_Horizon for the 'dates.start' of the Problem. The default, 0, will not exclude any OVERSIZE Problem.

Default: 0

<i>Extensions</i>	<i>BUCKET_OVERSIZE Extension</i>
-------------------	----------------------------------

**BUCKET\_OVERSIZE** -- *a category extension of model Problem\_Set*

A BUCKET\_OVERSIZE Problem\_Set selects BUCKET\_OVERSIZE Problems with certain characteristics.

**resource\_plan\_filter** -- *a Expression field of model BUCKET\_OVERSIZE*

An Expression that is passed a Resource\_Plan as '#' and should return "true" if the BUCKET\_OVERSIZE Problems on that Resource should be included in the Problem\_Set. If it returns "false", then Problems on that Resource will be excluded.

Default: true

**min\_bucket\_oversize** -- *a Percentage field of model BUCKET\_OVERSIZE*

A Problem will be excluded if its 'bucket\_oversize' is less than the 'min\_bucket\_oversize' The default, 0, will not exclude any BUCKET\_OVERSIZE Problem.

Default: 0

**UNDERLOAD** -- *a category extension of model Problem\_Set*

An UNDERLOAD Problem\_Set select UNDERLOAD Problems with certain characteristics.

**resource\_plan\_filter** -- *a Expression field of model UNDERLOAD*

An Expression that is passed a Resource\_Plan as '#' and should return "true" if the Underload Problems on that Resource should be included in the Problem\_Set. If it returns "false", then Problems on that Resource will be excluded.

Default: true

**min\_underload** -- *a Percentage field of model UNDERLOAD*

A Problem will be excluded if its 'underload' is less than the 'min\_underload' The default, 0, will not exclude any UNDERLOAD Problem.

Default: 0

**RESOURCE** -- *a category extension of model Problem\_Set*

A RESOURCE Problem\_Set selects Resource Problems (e.g., OVERLOAD, OVERSIZE, OVERTIME) on certain Resources.

resource\_plan\_filter - - a Expression field of model RESOURCE

An Expression that is passed a Resource\_Plan as '#' and should return "true" if the Problems on that Resource should be included in the Problem\_Set. If it returns "false", then Problems on that Resource will be excluded.  
Default: true

problem\_filter - - a Expression field of model RESOURCE

An Expression that is passed a Problem as '#' and should return "true" if the identified RESOURCE Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
Default: true

NEGATIVE\_ON\_HAND -- a category extension of model Problem\_Set

A NEGATIVE\_ON\_HAND Problem\_Set selects LOW\_ON\_HAND Problems with certain characteristics.

buffer\_plan\_filter - - a Expression field of model NEGATIVE\_ON\_HAND

An Expression that is passed a Buffer\_Plan as '#' and should return "true" if the NEGATIVE\_ON\_HAND Problems on that Buffer should be included in the Problem\_Set. If it returns "false", then Problems on that Buffer will be excluded.  
Default: true

problem\_filter - - a Expression field of model NEGATIVE\_ON\_HAND

An Expression that is passed a Problem as '#' and should return "true" if the identified NEGATIVE\_ON\_HAND Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
Default: true

min\_deficit - - a Quantity field of model NEGATIVE\_ON\_HAND

A Problem will be excluded if its 'deficit' is less than the minimum Quantity specified by this Quantity\_Horizon for the 'dates.start' of the Problem. The default, 0, will not exclude any NEGATIVE\_ON\_HAND Problem.  
Default: 0

Properties: command=True

OVER\_FLOW\_LIMIT -- a category extension of model Problem\_Set

An OVER\_FLOW\_LIMIT Problem\_Set selects OVER\_FLOW\_LIMIT Problems with certain characteristics.

buffer\_plan\_filter - - a Expression field of model OVER\_FLOW\_LIMIT

An Expression that is passed a Buffer\_Plan as '#' and should return "true" if the OVER\_FLOW\_LIMIT Problems on that Buffer should be included in the Problem\_Set. If it returns "false", then Problems on that Buffer will be excluded.  
Default: true

problem\_filter - - a Expression field of model OVER\_FLOW\_LIMIT

An Expression that is passed a Problem as '#' and should return "true" if the identified OVER\_FLOW\_LIMIT Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
Default: true

min\_deficit - - a Quantity field of model OVER\_FLOW\_LIMIT

A Problem will be excluded if its 'deficit' is less than the minimum Quantity specified by this Quantity\_Horizon for the 'dates.start' of the Problem. The default, 0, will not exclude any NEGATIVE\_ON\_HAND Problem.  
Default: 0

Properties: command=True

NEGATIVE\_ON\_HAND\_AT\_END -- a category extension of model Problem\_Set

A NEGATIVE\_ON\_HAND Problem\_Set selects LOW\_ON\_HAND Problems with certain characteristics.

buffer\_plan\_filter - - a Expression field of model NEGATIVE\_ON\_HAND\_AT\_END

An Expression that is passed a Buffer\_Plan as '#' and should return "true" if the NEGATIVE\_ON\_HAND Problems on that Buffer should be included in the Problem\_Set. If it returns "false", then Problems on that Buffer will be excluded.  
Default: true

problem\_filter - - a Expression field of model NEGATIVE\_ON\_HAND\_AT\_END

An Expression that is passed a Problem as '#' and should return "true" if the identified NEGATIVE\_ON\_HAND\_AT\_END Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
Default: true

min\_deficit - - a Quantity field of model NEGATIVE\_ON\_HAND\_AT\_END

A Problem will be excluded if its 'deficit' is less than the minimum Quantity specified by this Quantity\_Horizon for the 'dates.start' of the Problem. The default, 0, will not exclude any NEGATIVE\_ON\_HAND Problem.  
Default: 0

Extensions	LOT_OVER_CONSUMED Extension
------------	-----------------------------

Properties:    command=True

**LOT\_OVER\_CONSUMED** -- *a category extension of model Problem\_Set*

A LOT\_OVER\_CONSUMED Problem\_Set selects LOT\_OVER\_CONSUMED Problems with certain characteristics.

**buffer\_plan\_filter** -- *a Expression field of model LOT\_OVER\_CONSUMED*  
 An Expression that is passed a Buffer\_Plan as '#' and should return "true" if the LOT\_OVER\_CONSUMED Problems on that Buffer should be included in the Problem\_Set. If it returns "false", then Problems on that Buffer will be excluded.  
 Default:    true

**problem\_filter** -- *a Expression field of model LOT\_OVER\_CONSUMED*  
 An Expression that is passed a Problem as '#' and should return "true" if the identified LOT\_OVER\_CONSUMED Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default:    true

**LOT\_NOT\_CONSUMED** -- *a category extension of model Problem\_Set*

A LOT\_NOT\_CONSUMED Problem\_Set selects LOT\_NOT\_CONSUMED Problems with certain characteristics.

**buffer\_plan\_filter** -- *a Expression field of model LOT\_NOT\_CONSUMED*  
 An Expression that is passed a Buffer\_Plan as '#' and should return "true" if the LOT\_OVER\_CONSUMED Problems on that Buffer should be included in the Problem\_Set. If it returns "false", then Problems on that Buffer will be excluded.  
 Default:    true

**problem\_filter** -- *a Expression field of model LOT\_NOT\_CONSUMED*  
 An Expression that is passed a Problem as '#' and should return "true" if the identified LOT\_NOT\_CONSUMED Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default:    true

**LOT\_NOT\_PRODUCED** -- *a category extension of model Problem\_Set*

A LOT\_NOT\_PRODUCED Problem\_Set selects LOT\_NOT\_PRODUCED Problems with certain characteristics.

Extensions	LOT_OVER_PRODUCED Extension
------------	-----------------------------

**buffer\_plan\_filter** -- *a Expression field of model LOT\_NOT\_PRODUCED*  
 An Expression that is passed a Buffer\_Plan as '#' and should return "true" if the LOT\_OVER\_PRODUCED Problems on that Buffer should be included in the Problem\_Set. If it returns "false", then Problems on that Buffer will be excluded.  
 Default:    true

**problem\_filter** -- *a Expression field of model LOT\_NOT\_PRODUCED*  
 An Expression that is passed a Problem as '#' and should return "true" if the identified LOT\_NOT\_PRODUCED Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default:    true

**LOT\_OVER\_PRODUCED** -- *a category extension of model Problem\_Set*

A LOT\_OVER\_PRODUCED Problem\_Set selects LOT\_OVER\_PRODUCED Problems with certain characteristics.

**buffer\_plan\_filter** -- *a Expression field of model LOT\_OVER\_PRODUCED*  
 An Expression that is passed a Buffer\_Plan as '#' and should return "true" if the LOT\_OVER\_PRODUCED Problems on that Buffer should be included in the Problem\_Set. If it returns "false", then Problems on that Buffer will be excluded.  
 Default:    true

**problem\_filter** -- *a Expression field of model LOT\_OVER\_PRODUCED*  
 An Expression that is passed a Problem as '#' and should return "true" if the identified LOT\_OVER\_PRODUCED Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default:    true

**min\_excess\_time** -- *a Time field of model LOT\_OVER\_PRODUCED*  
 Problems with less excess time than 'min\_excess\_time' between supplier completion date and consumer start date are excluded. This field provides a filter that can be used to allow strategy ignore small allowable excess times between consumer and supplier.  
 Default:    0

**LOW\_ON\_HAND** -- *a category extension of model Problem\_Set*

A LOW\_ON\_HAND Problem\_Set selects LOW\_ON\_HAND Problems with certain characteristics.

<i>Extensions</i>	<i>EXCESS_ON_HAND Extension</i>
-------------------	---------------------------------

**buffer\_plan\_filter** - - *a Expression field of model LOW\_ON\_HAND*  
 An Expression that is passed a Buffer\_Plan as '#' and should return "true" if the LOW\_ON\_HAND Problems on that Buffer should be included in the Problem\_Set. If it returns "false", then Problems on that Buffer will be excluded.  
 Default: true

**problem\_filter** - - *a Expression field of model LOW\_ON\_HAND*  
 An Expression that is passed a Problem as '#' and should return "true" if the identified LOW\_ON\_HAND Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default: true

**min\_deficit** - - *a Quantity field of model LOW\_ON\_HAND*  
 A Problem will be excluded if its 'deficit' is less than the minimum Quantity specified by this Quantity\_Horizon for the 'dates.start' of the Problem. The default, 0, will not exclude any LOW\_ON\_HAND Problem.  
 Default: 0

**EXCESS\_ON\_HAND** -- *a category extension of model Problem\_Set*  
 An EXCESS\_ON\_HAND Problem\_Set selects EXCESS\_ON\_HAND Problems with certain characteristics.

**buffer\_plan\_filter** - - *a Expression field of model EXCESS\_ON\_HAND*  
 An Expression that is passed a Buffer\_Plan as '#' and should return "true" if the EXCESS\_ON\_HAND Problems on that Buffer should be included in the Problem\_Set. If it returns "false", then Problems on that Buffer will be excluded.  
 Default: true

**problem\_filter** - - *a Expression field of model EXCESS\_ON\_HAND*  
 An Expression that is passed a Problem as '#' and should return "true" if the identified EXCESS\_ON\_HAND Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default: true

**min\_excess** - - *a Quantity field of model EXCESS\_ON\_HAND*  
 A Problem will be excluded if its 'excess' is less than the minimum Quantity specified by this Quantity\_Horizon for the 'dates.start' of the Problem. The default, 0, will not exclude any EXCESS\_ON\_HAND Problem.  
 Default: 0

<i>Extensions</i>	<i>EXCESS_ON_HAND_AT_END Extension</i>
-------------------	--

**EXCESS\_ON\_HAND\_AT\_END** -- *a category extension of model Problem\_Set*

An EXCESS\_ON\_HAND\_AT\_END Problem\_Set selects EXCESS\_ON\_HAND\_AT\_END Problems with certain characteristics.

**buffer\_plan\_filter** - - *a Expression field of model EXCESS\_ON\_HAND\_AT\_END*  
 An Expression that is passed a Buffer\_Plan as '#' and should return "true" if the EXCESS\_ON\_HAND\_AT\_END Problems on that Buffer should be included in the Problem\_Set. If it returns "false", then Problems on that Buffer will be excluded.  
 Default: true

**problem\_filter** - - *a Expression field of model EXCESS\_ON\_HAND\_AT\_END*  
 An Expression that is passed a Problem as '#' and should return "true" if the identified EXCESS\_ON\_HAND\_AT\_END Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default: true

**min\_excess** - - *a Quantity field of model EXCESS\_ON\_HAND\_AT\_END*  
 A Problem will be excluded if its 'excess' is less than the minimum Quantity specified by this Quantity\_Horizon for the 'dates.start' of the Problem. The default, 0, will not exclude any EXCESS\_ON\_HAND\_AT\_END Problem.  
 Default: 0

**BUFFER** -- *a category extension of model Problem\_Set*

A BUFFER Problem\_Set selects Buffer Problems (e.g., LOW\_ON\_HAND, EXCESS\_ON\_HAND, etc.) on certain Buffers.

**buffer\_plan\_filter** - - *a Expression field of model BUFFER*  
 An Expression that is passed a Buffer\_Plan as '#' and should return "true" if the Problems on that Buffer should be included in the Problem\_Set. If it returns "false", then Problems on that Buffer will be excluded.  
 Default: true

**problem\_filter** - - *a Expression field of model BUFFER*  
 An Expression that is passed a Problem as '#' and should return "true" if the identified BUFFER Problems should be included in the Problem\_Set. If it returns "false", then the identified Problems will be excluded.  
 Default: true



<i>Extensions</i>	<i>Strategy_Change Extensions</i>
-------------------	-----------------------------------

### 10.38 Strategy\_Change Extensions

10.38.1 change\_category extensions of model Strategy\_Change

MOVE\_IN -- *a change\_category extension of model Strategy\_Change*

MOVE\_IN strategy change

MOVE\_OUT -- *a change\_category extension of model Strategy\_Change*

MOVE\_OUT strategy change

SPLIT -- *a change\_category extension of model Strategy\_Change*

Split strategy change

USE\_MORE -- *a change\_category extension of model Strategy\_Change*

Use\_More strategy change

USE\_LESS -- *a change\_category extension of model Strategy\_Change*

Use\_Less strategy change

USE\_ALTERNATE\_OPERATION -- *a change\_category extension of model Strategy\_Change*

Use\_Alternate\_Operation strategy change

min\_alt -- *a Integer field of model USE\_ALTERNATE\_OPERATION*  
 The min\_alt and max\_alt defines the range of alternate operations considered. For example, if min\_alt = max\_alt = 0, only the first alternate is allowed.  
 Default: 0

max\_alt -- *a Integer field of model USE\_ALTERNATE\_OPERATION*  
 See notes in min\_alt.  
 Default: INT\_MAX

USE\_ALTERNATE\_RESOURCE -- *a change\_category extension of model Strategy\_Change*

Use\_Alternate\_Resource strategy change

<i>Extensions</i>	<i>USE_EFFECTIVE_ALTERNATE Extension</i>
-------------------	--

USE\_EFFECTIVE\_ALTERNATE -- *a change\_category extension of model Strategy\_Change*

Use\_Effective\_Alternate strategy change

INCREASE\_CAPACITY -- *a change\_category extension of model Strategy\_Change*

Increase\_Capacity strategy change

DECREASE\_CAPACITY -- *a change\_category extension of model Strategy\_Change*

Decrease\_Capacity strategy change

RESIZE\_MORE -- *a change\_category extension of model Strategy\_Change*

Resize\_More strategy change

RESIZE\_LESS -- *a change\_category extension of model Strategy\_Change*

Resize\_Less strategy change

UPSTREAM -- *a change\_category extension of model Strategy\_Change*

Upstream strategy change

DOWNSTREAM -- *a change\_category extension of model Strategy\_Change*

Downstream strategy change

10.39 Strategy\_Lock Extensions

10.39.1 spec extensions of model Strategy\_Lock

OPERATION\_PLAN\_RANK\_RANGE -- a spec extension of model Strategy\_Lock

The OPERATION\_PLAN\_RANK\_RANGE 'spec' extension allows users to specify a range which will lock Operation\_Plan's whose 'rank' lies within it.

min\_rank -- a Number field of model OPERATION\_PLAN\_RANK\_RANGE  
Specifies the min of the range, within which Operation\_Plan's will be locked.  
Default: -oo

max\_rank -- a Number field of model OPERATION\_PLAN\_RANK\_RANGE  
Specifies the max of the range, within which Operation\_Plan's will be locked.  
Default: oo

OPERATION\_PLAN\_RANK\_EXPRESSION -- a spec extension of model Strategy\_Lock

The OPERATION\_PLAN\_RANK\_EXPRESSION 'spec' extension specifies an Expression that is passed an Operation\_Plan as '#' and should return "true" if the given Operation\_Plan should be considered locked.

expression -- a Expression field of model  
OPERATION\_PLAN\_RANK\_EXPRESSION  
An Expression that is passed an Operation\_Plan as '#' and that Operation\_Plan will be locked if it returns "true".  
Default: false

10.40 Strategy\_Goal Extensions

10.40.1 goal extensions of model Strategy\_Goal

FEASIBILITY -- a goal extension of model Strategy\_Goal

The goal is to minimize the sum of 'interaction' of infeasible Problems, ultimately driving it to zero by eliminating all infeasible Problems.

target\_interaction -- a Number field of model FEASIBILITY  
The target level for the sum of 'interaction' of all infeasible Problems. If that sum is less than or equal to target\_interaction, then the target has been met (though the goal to minimize it to zero still remains).

This is typically used in conjunction with target-oriented 'termination' extensions.  
Default: 0

MINIMIZE\_PROBLEM\_COUNT -- a goal extension of model Strategy\_Goal

The goal is to minimize the number of Problems, ultimately driving it to zero by eliminating all Problems. Most applications should use MINIMIZE\_PROBLEMS instead of MINIMIZE\_PROBLEM\_COUNT.

target\_problem\_count -- a Number field of model  
MINIMIZE\_PROBLEM\_COUNT  
The target level for the number of Problems. If that number is less than or equal to target\_problem\_count, then the target has been met (though the goal to minimize it to zero still remains).

This is typically used in conjunction with target-oriented 'termination' extensions.  
Default: 0

MINIMIZE\_PROBLEMS -- a goal extension of model Strategy\_Goal

The goal is to minimize problems in all problem sets. This goal differs from MINIMIZE\_PROBLEM\_COUNT in that problems are internally assigned a weight based on their size and importance, and the strategy tries to minimize the sum of these weights (instead of the number of problems).

target\_problems - - a Number field of model MINIMIZE\_PROBLEMS

The target level for the sum of the weights of all problems. If that number is less than or equal to target\_problems, then the target has been met (though the goal to minimize it to zero still remains).

This is typically used in conjunction with target-oriented 'termination' extensions.  
Default: 0

MINIMIZE\_LATENCY - - a goal extension of model Strategy\_Goal

The goal is to minimize the sum of 'latency' of all item requests.

target\_latency - - a Number field of model MINIMIZE\_LATENCY

The target level for the sum of 'latency' of all requests. If that sum is less than or equal to target\_latency, then the target has been met (though the goal to minimize it to zero still remains).

This is typically used in conjunction with target-oriented 'termination' extensions.  
Default: 0

WEIGHTED\_LATENCY - - a goal extension of model Strategy\_Goal

The goal is to minimize the weighted sum of 'latency' of all item requests.

target\_latency - - a Number field of model WEIGHTED\_LATENCY

The target level for the sum of 'latency' of all requests. If that sum is less than or equal to target\_latency, then the target has been met (though the goal to minimize it to zero still remains).

This is typically used in conjunction with target-oriented 'termination' extensions.  
Default: 0

day\_latency - - a Number field of model WEIGHTED\_LATENCY

The weight for late order. Weighted latency equals days late raised to day\_latency times quantity late raised to quantity\_latency.  
Default: 1.0

quantity\_latency - - a Number field of model WEIGHTED\_LATENCY

The weight for late order. Weighted latency equals days late raised to day\_latency times quantity late raised to quantity\_latency.  
Default: 0.0

MINIMIZE\_SHORTNESS - - a goal extension of model Strategy\_Goal

The goal is to minimize the sum of 'shortness' of all item requests.

target\_shortness - - a Number field of model MINIMIZE\_SHORTNESS

The target level for the sum of 'shortness' of all requests. If that sum is less than or equal to target\_shortness, then the target has been met (though the goal to minimize it to zero still remains).

This is typically used in conjunction with target-oriented 'termination' extensions.  
Default: 0

WEIGHTED\_SHORTNESS - - a goal extension of model Strategy\_Goal

The goal is to minimize the weighted sum of 'shortness' of all item requests.

target\_shortness - - a Number field of model WEIGHTED\_SHORTNESS

The target level for the sum of 'shortness' of all requests. If that sum is less than or equal to target\_shortness, then the target has been met (though the goal to minimize it to zero still remains).

This is typically used in conjunction with target-oriented 'termination' extensions.  
Default: 0

quantity\_short\_power - - a Number field of model WEIGHTED\_SHORTNESS

The weight for short order. Weighted shortness equals quantity short raised to quantity\_short\_power.  
Default: 1.0

MINIMIZE\_COST - - a goal extension of model Strategy\_Goal

The goal is to minimize the total 'cost' of the plan.

target\_cost - - a Number field of model MINIMIZE\_COST

The target level for 'cost'. If 'cost' is less than or equal to target\_cost, then the target has been met (though the goal to minimize cost to zero still remains). This is typically used in conjunction with target-oriented 'termination' extensions.  
Default: 0

MAXIMIZE\_PROFIT - - a goal extension of model Strategy\_Goal

<i>Extensions</i>	<i>MAXIMIZE_REVENUE Extension</i>
-------------------	-----------------------------------

The goal is to maximize the difference between 'revenue' and 'cost'. Revenue is currently taken to be the sum of the planned price of the item requests in the committed forecast that are satisfied by the current plan.

*target\_profit* -- a Number field of model MAXIMIZE\_PROFIT

The target level for 'profit'. If 'profit' is greater than or equal to 'target\_profit', then the target has been met (though the goal to maximize it still remains).

This is typically used in conjunction with target-oriented 'termination' extensions.  
Default: 0

#### MAXIMIZE\_REVENUE -- a goal extension of model Strategy\_Goal

The goal is to maximize 'revenue'. Revenue is currently taken to be the sum of the planned price of the item requests in the committed forecast that are satisfied by the current plan.

*target\_revenue* -- a Number field of model MAXIMIZE\_REVENUE

The target level for 'revenue'. If 'revenue' is greater than or equal to 'target\_revenue', then the target has been met (though the goal to maximize it still remains).

This is typically used in conjunction with target-oriented 'termination' extensions.  
Default: 0

<i>Extensions</i>	<i>Calendar_Entry Extensions</i>
-------------------	----------------------------------

### 10.41 Calendar\_Entry Extensions

#### 10.41.1 value extensions of model Calendar\_Entry

##### NUMBER -- a value extension of model Calendar\_Entry

Entry specifies a single Number.

*number* -- a Number field of model NUMBER  
Number associated with this entry  
Default: 0

##### QUANTITY -- a value extension of model Calendar\_Entry

Entry specifies a single Quantity.

*quantity* -- a Quantity field of model QUANTITY  
Quantity associated with this entry  
Default: 0 (unitless)

##### NUMBER\_QUANTITY -- a value extension of model Calendar\_Entry

Entry specifies a Number and a Quantity.

*number* -- a Number field of model NUMBER\_QUANTITY  
Number associated with this entry  
Default: 0

*quantity* -- a Quantity field of model NUMBER\_QUANTITY  
Quantity associated with this entry  
Default: 0 (unitless)

##### SYMBOL -- a value extension of model Calendar\_Entry

Entry specifies a single Symbol.

*symbol* -- a Symbol field of model SYMBOL  
Symbol associated with this entry  
Default: none

##### TIME -- a value extension of model Calendar\_Entry

Entry specifies a single Time

<i>Extensions</i>	<i>day_pattern extensions of model Calendar_Entry</i>
-------------------	---

time - - *a Time field of model TIME*  
Time associated with this entry  
Default: 0

#### 10.41.2 day\_pattern extensions of model Calendar\_Entry

##### EVERYDAY -- *a day\_pattern extension of model Calendar\_Entry*

An EVERYDAY Calendar\_Entry extension is applicable to every single day, or Date, within the Calendar\_Entry's 'effective' range.

##### EVERY\_N\_DAYS -- *a day\_pattern extension of model Calendar\_Entry*

An EVERY\_N\_DAYS Calendar\_Entry extension is applicable to every 'nth' day, or Date, within the Calendar\_Entry's 'effective' range, where the 'effective.start' Date is the first applicable day. For instance, every 15 days between Jan. 1, 1996 and Jul. 31, 1996. If 'effective.start' is "oo\_past", or "oo\_future", then the Calendar\_Entry does not apply to any day.

##### nth - - *a Integer field of model EVERY\_N\_DAYS*

The interval between any applicable day and the next applicable day. If 'nth' is 15, and 'effective.start' is "Jan 1", then the applicable days are: Jan. 1, Jan. 16, Jan. 31, Feb. 15, ... The default is 1, which is equivalent to the EVERYDAY Calendar\_Entry extension.  
Default: 1

##### WEEKDAYS -- *a day\_pattern extension of model Calendar\_Entry*

A WEEKDAYS Calendar\_Entry extension applies to all weekdays (Monday through Friday) in the 'effective' range of the Calendar\_Entry. See the WEEKENDS extension for the inverse.

##### WEEKENDS -- *a day\_pattern extension of model Calendar\_Entry*

A WEEKENDS Calendar\_Entry extension applies to all weekends (Saturday and Sunday) in the 'effective' range of the Calendar\_Entry. See the WEEKDAYS extension for the inverse.

##### DAYS\_OF\_WEEK -- *a day\_pattern extension of model Calendar\_Entry*

<i>Extensions</i>	<i>DAY_OF_MONTH Extension</i>
-------------------	-------------------------------

A DAYS\_OF\_WEEK Calendar\_Entry extension applies to the specified days of the week that are in the Calendar\_Entry's 'effective' range. For instance, every Monday, Wednesday, and Friday between Jan. 1, 1996 and July 31, 1996.

##### days - - *a String field of model DAYS\_OF\_WEEK*

A list that specifies the applicable days of the week for this Calendar\_Entry. Entries in the list are separated by commas or white space. Each element of the list may be a single day (e.g., Mon), or a range of days (e.g., Mon-Fri). The following are valid:

"Mo,Tu,We", "Sa,Su", "Monday-Friday", "Tu,Th", or "Mo-Tu,Th Fr,Sa". Note that "Mo-Fr" is equivalent to the WEEKDAYS extension, and "Sa,Su" is equivalent to the WEEKENDS extension. If no days are specified, the default is "Mo-Su" which is equivalent to the EVERYDAY extension.

Because the names of the week days change in internationalized versions, be sure to specify enough characters of each day to be non-ambiguous. For example, in English, "M-F" is not ambiguous, while "M,T,W,T,F" is ambiguous. The latter should be specified as "M,Tu,W,Th,F" to avoid any ambiguity.  
Default: Mo-Su

##### DAY\_OF\_MONTH -- *a day\_pattern extension of model Calendar\_Entry*

A DAY\_OF\_MONTH Calendar\_Entry extension applies to a specific day in a month, a subset of months, or all months. For example, the 15th day of each month, the 25th day of December, the first day of each quarter (Jan, Apr, Jul, Oct), the -1th day of each month (the last day of each month), and so on, are examples.

##### day - - *a Integer field of model DAY\_OF\_MONTH*

The day of month on which this Calendar\_Entry is applicable. Valid values are [-31] inclusive which represent the 1st, 2nd, 3rd, etc. days of the month. The default is the first day of the month (1).  
Default: 1

##### months - - *a String field of model DAY\_OF\_MONTH*

A list that specifies the months in which this Calendar\_Entry is applicable. Entries in the list may be separated by commas or white space. Each element of the list may be a single month (e.g., January), a range of months (e.g., May-Aug). The following are valid: "J,F,Mar", "Jun-Jul", "May-Aug" or "Jan-Mar,Jun-Nov,Dec". At least the first 3 characters of the name of each month must be specified. The default is "Jan-Dec" which specifies all months.

Extensions	DAY_OF_WEEK_OF_MONTH Extension
------------	--------------------------------

Because the names of the months change in internationalized versions, be sure to specify enough characters of each month to be non-ambiguous. For example, in English, "J-D" is not ambiguous, while "J,F,M,A,M,J" is ambiguous. The latter should be specified as "J,F,Mar,Ap,May,Jun" to avoid any ambiguity.  
 Default: Jan-Dec

DAY\_OF\_WEEK\_OF\_MONTH -- a day\_pattern extension of model Calendar\_Entry

A DAY\_OF\_WEEK\_OF\_MONTH Calendar\_Entry pattern applies to the nth occurrence of a specific day of week of a month, a subset of months, or all months. For example, the 4th Thursday in November, the 1st Tuesday in Oct-Dec, the 1st Monday in September, or the 3rd Wednesday of each month.

day - - a Integer field of model DAY\_OF\_WEEK\_OF\_MONTH  
 The day of the week on which this Calendar\_Entry is applicable. Valid values are [1-7] where the value of 1 is Monday, 2 is Tuesday, ..., and 7 is Sunday. The default is Monday.  
 Default: 1

nth - - a Integer field of model DAY\_OF\_WEEK\_OF\_MONTH  
 Specifies the nth occurrence of the day of week on which this Calendar\_Entry is applicable. Valid values are [1-5] where the value of 1 is the 1st occurrence, 2 is the 2nd occurrence, etc. For example, the 4th Thursday, or the 1st Friday. The default is the first occurrence.  
 Default: 1

week - - a Integer field of model DAY\_OF\_WEEK\_OF\_MONTH  
 This field is obsolete and will be removed in 3\_06\_BETA Use 'nth' instead. Refer to 'nth' for documentation.  
 Default: 1  
 Properties: obsolete=True

months - - a String field of model DAY\_OF\_WEEK\_OF\_MONTH  
 A list that specifies the months in which this Calendar\_Entry is applicable. Entries in the list may be separated by commas or white space. Each element of the list may be a single month (e.g., January), a range of months (e.g., May-Aug). The following are valid: "J,F,Mar", "Jun Jul", "May-Aug" or "Jan-Mar,Jun NovDec". At least the first 3 characters of the name of each month must be specified. The default is "Jan-Dec" which specifies all months.

Extensions	DAY_OF_LAST_WEEK_OF_MONTH Extension
------------	-------------------------------------

Because the names of the months change in internationalized versions, be sure to specify enough characters of each month to be non-ambiguous. For example, in English, "J-D" is not ambiguous, while "J,F,M,A,M,J" is ambiguous. The latter should be specified as "J,F,Mar,Ap,May,Jun" to avoid any ambiguity.  
 Default: Jan-Dec

DAY\_OF\_LAST\_WEEK\_OF\_MONTH -- a day\_pattern extension of model Calendar\_Entry

A DAY\_OF\_LAST\_WEEK\_OF\_MONTH Calendar\_Entry applies to a specific day in the last week of a month, a subset of months, or all months. For example, the last Monday of May or the last Friday of each month.

day - - a Integer field of model DAY\_OF\_LAST\_WEEK\_OF\_MONTH  
 The day of the week on which this Calendar\_Entry is applicable. Valid values are [1-7] where the value of 1 is Monday, 2 is Tuesday, ..., and 7 is Sunday. The default is Monday.  
 Default: 1

months - - a String field of model DAY\_OF\_LAST\_WEEK\_OF\_MONTH  
 A list that specifies the months in which this Calendar\_Entry is applicable. Entries in the list may be separated by commas or white space. Each element of the list may be a single month (e.g., January), a range of months (e.g., May-Aug). The following are valid: "J,F,Mar", "Jun Jul", "May-Aug" or "Jan-Mar,Jun NovDec". At least the first 3 characters of the name of each month must be specified. The default is "Jan-Dec" which specifies all months.

Because the names of the months change in internationalized versions, be sure to specify enough characters of each month to be non-ambiguous. For example, in English, "J-D" is not ambiguous, while "J,F,M,A,M,J" is ambiguous. The latter should be specified as "J,F,Mar,Ap,May,Jun" to avoid any ambiguity.  
 Default: Jan-Dec

DAY\_OF\_YEAR -- a day\_pattern extension of model Calendar\_Entry

A DAY\_OF\_YEAR day\_pattern Calendar\_Entry applies to the Nth day of each year. For instance the 183rd day of the year specifies the mid-point of the year.

day - - a Integer field of model DAY\_OF\_YEAR  
 The day of the year on which this Calendar\_Entry is applicable. Valid values are [1-366] inclusive where the value of 1 is Jan 1. The default is Jan 1.  
 Default: 1

<i>Extensions</i>	<i>YEARLY Extension</i>
-------------------	-------------------------

**YEARLY** -- *a day\_pattern extension of model Calendar\_Entry*

A YEARLY day\_pattern Calendar\_Entry applies to a specific day in a specific month of each year. For instance, the 25th of December, the 4th of July, and the first of January.

**month** -- *a Integer field of model YEARLY*

Specifies the month of year in which this Calendar\_Entry is applicable. Valid values are [1-12] inclusive, where the value of 1 is January, 2 is February, ... , and 12 is December. The default is January.

Default: 1

**day** -- *a Integer field of model YEARLY*

Specifies the day of month on which this Calendar\_Entry is applicable. Valid values are [1-31] which represent the 1st, 2nd, 3rd, etc. days of the month. The default is the first day of the month (1).

Default: 1

<i>Extensions</i>	<i>Calendar Extensions</i>
-------------------	----------------------------

## 10.42 Calendar Extensions

**10.42.1 entry\_value extensions of model Calendar**

**UNSPECIFIED** -- *a entry\_value extension of model Calendar*

Calendar's entry\_value is not specified

**NUMBER** -- *a entry\_value extension of model Calendar*

Calendar's entries specify a single Number.

**default\_number** -- *a Number field of model NUMBER*

Default Number value for calendar This will be the value of this Calendar at any time in which no value was specified by the Calendar Entries. The default value of default\_number is 0.

Default: 0

**QUANTITY** -- *a entry\_value extension of model Calendar*

Calendar's entries specify a single Number.

**default\_quantity** -- *a Quantity field of model QUANTITY*

Default Quantity value for calendar This will be the value of this Calendar at any time in which no value was specified by the Calendar Entries. The default value of default\_quantity is 0 (unitless)

Default: 0 (unitless)

**NUMBER\_QUANTITY** -- *a entry\_value extension of model Calendar*

Calendar's entries specify a single Number\_Quantity\_Pair

**default\_number** -- *a Number field of model NUMBER\_QUANTITY*

Default Number value of NUMBER\_QUANTITY This will be the value of this Calendar at any time in which no value was specified by the Calendar Entries. The default value of default\_number is 0.

Default: 0

**default\_quantity** -- *a Quantity field of model NUMBER\_QUANTITY*

Default Quantity value of NUMBER\_QUANTITY This will be the value of this Calendar at any time in which no value was specified by the Calendar Entries. The default value of default\_quantity is 0 (unitless)

Default: 0 (unitless)

Extensions	SYMBOL Extension
------------	------------------

**SYMBOL** -- *a entry\_value extension of model Calendar*

Calendar's entries specify a single Symbol

*default\_value* -- *a Symbol field of model SYMBOL*

Default Symbol value for calendar This will be the value of this Calendar at any time in which no value was specified by the Calendar Entries. There is no default value of this default\_value.

Default: none

**TIME** -- *a entry\_value extension of model Calendar*

Calendar's entries specify a single Number.

*default\_time* -- *a Time field of model TIME*

Default Time value for calendar This will be the value of this Calendar at any time in which no value was specified by the Calendar Entries. The default value of default\_time is 0.0.  
Default: 0.0

Extensions	Flow_Criterion Extensions
------------	---------------------------

### 10.43 Flow\_Criterion Extensions

10.43.1 criterion extensions of model Flow\_Criterion

**CUSTOMER\_RANK** -- *a criterion extension of model Flow\_Criterion*

A CUSTOMER\_RANK Flow\_Criterion uses the 'rank' field of the customer Site (delivery\_request.customer\_plan.site.rank) as a relative weighting or sort criterion.

*produce\_separately* -- *a Logical field of model CUSTOMER\_RANK*

If "true", then when generating producing Operation\_Plans, it will generate a separate Operation\_Plan in each bucket for each different customer rank of consuming Flow\_Plan.

Default: false

*default\_rank* -- *a Number field of model CUSTOMER\_RANK*

If there is no customer (or no Delivery\_Request) associated with the Flow\_Plan, then this default\_rank is used instead of the customer rank.  
Default: 0

**SELLER\_RANK** -- *a criterion extension of model Flow\_Criterion*

A SELLER\_RANK Flow\_Criterion uses the 'rank' field of the Seller as a relative weighting or sort criterion.

*produce\_separately* -- *a Logical field of model SELLER\_RANK*

If "true", then when generating producing Operation\_Plans, it will generate a separate Operation\_Plan in each bucket for each different Seller rank of consuming Flow\_Plan.

Default: false

*default\_rank* -- *a Number field of model SELLER\_RANK*

If there is no Seller (or no Delivery\_Request) associated with the Flow\_Plan, then this default\_rank is used instead of the customer rank.  
Default: 0

**REQUEST\_RANK** -- *a criterion extension of model Flow\_Criterion*

A REQUEST\_RANK Flow\_Criterion uses the 'rank' field of the Delivery\_Request, as ranked by its 'customer\_plan' as a relative weighting or sort criterion.



produce\_separately - - *a Logical field of model REQUEST\_RANK*  
If "true", then when generating producing Operation\_Plans, it will generate a separate Operation\_Plan in each bucket for each different Delivery\_Request rank of consuming Flow\_Plan.  
Default: false

default\_rank - - *a Number field of model REQUEST\_RANK*  
If there is no Delivery\_Request associated with the Flow\_Plan, then this 'default\_rank' is used instead of the customer rank.  
Default: 0

**ACTUAL\_OR\_FORECAST** - - *a criterion extension of model Flow\_Criterion*

A ACTUAL\_OR\_FORECAST Flow\_Criterion uses the Number 1 if the Flow\_Plan is associated with an actual Delivery\_Request, the Number 0 if the Flow\_Plan is associated with a forecast Delivery\_Request, or the 'default\_rank' if the Flow\_Plan is not associated with a Delivery\_Request, as a relative weighing or sort criterion.

produce\_separately - - *a Logical field of model ACTUAL\_OR\_FORECAST*  
If "true", then when generating producing Operation\_Plans, it will generate a separate Operation\_Plan in each bucket for actuals, forecasts, and other consuming Flow\_Plans.  
Default: false

default\_rank - - *a Number field of model ACTUAL\_OR\_FORECAST*  
If there is no Delivery\_Request associated with the Flow\_Plan, then this 'default\_rank' is used instead of 1 or 0.  
Default: 0

**REQUEST\_ISSUED** - - *a criterion extension of model Flow\_Criterion*

A REQUEST\_ISSUED Flow\_Criterion uses the 'issued' field of the Request as a relative weighing or sort criterion.

produce\_separately - - *a Logical field of model REQUEST\_ISSUED*  
If "true", then when generating producing Operation\_Plans, it will generate a separate Operation\_Plan in each bucket for each different customer rank of consuming Flow\_Plan.  
Default: false

default\_rank - - *a Number field of model REQUEST\_ISSUED*  
If there is no customer (or no Delivery\_Request) associated with the Flow\_Plan, then this 'default\_rank' is used instead of the customer rank.  
Default: 0

default\_issued - - *a Date field of model REQUEST\_ISSUED*  
If there is no Request associated with the Flow\_Plan, then this 'default\_issued' Date is used instead.  
Default: 0

**PROMISE\_DUE** - - *a criterion extension of model Flow\_Criterion*

A PROMISE\_DUE Flow\_Criterion uses the 'due.end' field of the Delivery\_Promise as a relative weighing or sort criterion.

produce\_separately - - *a Logical field of model PROMISE\_DUE*  
If "true", then when generating producing Operation\_Plans, it will generate a separate Operation\_Plan in each bucket for each different customer rank of consuming Flow\_Plan.  
Default: false

default\_rank - - *a Number field of model PROMISE\_DUE*  
If there is no customer (or no Delivery\_Request) associated with the Flow\_Plan, then this 'default\_rank' is used instead of the customer rank.  
Default: 0

default\_due - - *a Date field of model PROMISE\_DUE*  
If there is no Delivery\_Promise associated with the Flow\_Plan, then this 'default\_due' Date is used instead.  
Default: 0

**REQUEST\_DUE** - - *a criterion extension of model Flow\_Criterion*

A REQUEST\_DUE Flow\_Criterion uses the 'r\_due.end' field of the Delivery\_Promise as a relative weighing or sort criterion.

produce\_separately - - *a Logical field of model REQUEST\_DUE*  
If "true", then when generating producing Operation\_Plans, it will generate a separate Operation\_Plan in each bucket for each different customer rank of consuming Flow\_Plan.  
Default: false

**default\_rank** -- a Number field of model REQUEST\_DUE  
If there is no customer (or no Delivery\_Request) associated with the Flow\_Plan, then this default\_rank is used instead of the customer rank.  
Default: 0

**default\_due** -- a Date field of model REQUEST\_DUE  
If there is no Delivery\_Promise associated with the Flow\_Plan, then this default\_due Date is used instead.  
Default: 0

**DUE\_DATE** -- a criterion extension of model Flow\_Criterion

A DUE\_DATE Flow\_Criterion uses the accepted due date, if it exists, or the promise due date otherwise, as a relative weighting or sort criterion.

**produce\_separately** -- a Logical field of model DUE\_DATE  
If "true", then when generating producing Operation\_Plans, it will generate a separate Operation\_Plan in each bucket for each different customer rank of consuming Flow\_Plan.  
Default: false

**default\_rank** -- a Number field of model DUE\_DATE  
If there is no customer (or no Delivery\_Request) associated with the Flow\_Plan, then this default\_rank is used instead of the customer rank.  
Default: 0

**default\_due** -- a Date field of model DUE\_DATE  
If there is no Delivery\_Promise associated with the Flow\_Plan, then this default\_due Date is used instead.  
Default: 0

**PROMISED** -- a criterion extension of model Flow\_Criterion

A PROMISED Flow\_Criterion uses the Number 1 if the Flow\_Plan is associated with a Delivery\_Promise with due\_end Date in the same bucket as the 'due\_end' Date of the Delivery\_Request, the Number 0 if they are in different buckets, or the default\_rank if the Flow\_Plan is not associated with a Delivery\_Request, as a relative weighting or sort criterion.

**produce\_separately** -- a Logical field of model PROMISED  
If "true", then when generating producing Operation\_Plans, it will generate a separate Operation\_Plan in each bucket for each different ranking (1, 0, or default\_rank) of consuming Flow\_Plan.  
Default: false

**default\_rank** -- a Number field of model PROMISED  
If there is no Delivery\_Request associated with the Flow\_Plan, then this default\_rank is used instead of 1 or 0.  
Default: 0

**ENTRY\_DATE** -- a criterion extension of model Flow\_Criterion

An ENTRY\_DATE Flow\_Criterion uses the order entry date as a relative weighting or sort criterion.

**produce\_separately** -- a Logical field of model ENTRY\_DATE  
If "true", then when generating producing Operation\_Plans, it will generate a separate Operation\_Plan in each bucket for each different customer rank of consuming Flow\_Plan.  
Default: false

**default\_rank** -- a Number field of model ENTRY\_DATE  
If there is no customer (or no Delivery\_Request) associated with the Flow\_Plan, then this default\_rank is used instead of the customer rank.  
Default: 0

10.44 Calendar\_Plan Extensions

10.44.1 entry\_value extensions of model Calendar\_Plan

NUMBER -- a entry\_value extension of model Calendar\_Plan

A NUMBER Calendar\_Plan contains a Number for each Date. A NUMBER Calendar dictates the values during the horizon', and the before\_horizon\_number' and after\_horizon\_number' fields specify the values used beyond the horizon'.

before\_horizon\_number -- a Number field of model NUMBER

This will be the value of this Calendar\_Plan at any Date before the horizon'.

This is not settable. It is dictated by the model that provides the Calendar\_Plan field. See the documentation for the Calendar\_Plan field of interest.

Properties: Export-Only Field

after\_horizon\_number -- a Number field of model NUMBER

This will be the value of this Calendar\_Plan at any Date after the horizon'.

This is not settable. It is dictated by the model that provides the Calendar\_Plan field. See the documentation for the Calendar\_Plan field of interest.

Properties: Export-Only Field

intra\_horizon\_number -- a Number field of model NUMBER

This will be the value of this Calendar\_Plan at any Date within the horizon' where no other Calendar\_Entry is active. If not set, the default from 'calendar' is used.

Default: 0

QUANTITY -- a entry\_value extension of model Calendar\_Plan

A QUANTITY Calendar\_Plan contains a Quantity for each Date. A QUANTITY Calendar dictates the values during the horizon', and the before\_horizon\_quantity' and after\_horizon\_quantity' fields specify the values used beyond the horizon'.

Note that currently only unitless Quantities are supported. In the future, the QUANTITY Calendar will have a 'unit' field which defines the Unit to which all the entries' quantities are converted.

before\_horizon\_quantity -- a Quantity field of model QUANTITY

This will be the value of this Calendar\_Plan at any Date before the horizon'.

This is not settable. It is dictated by the model that provides the Calendar\_Plan field. See the documentation for the Calendar\_Plan field of interest.

Properties: Export-Only Field

after\_horizon\_quantity -- a Quantity field of model QUANTITY

This will be the value of this Calendar\_Plan at any Date after the horizon'.

This is not settable. It is dictated by the model that provides the Calendar\_Plan field. See the documentation for the Calendar\_Plan field of interest.

Properties: Export-Only Field

intra\_horizon\_quantity -- a Quantity field of model QUANTITY

This will be the value of this Calendar\_Plan at any Date within the horizon' where no other Calendar\_Entry is active. If not set, the default from 'calendar' is used.

Default: 0 (unitless)

NUMBER\_QUANTITY -- a entry\_value extension of model Calendar\_Plan

A NUMBER\_QUANTITY Calendar\_Plan contains a Number and a Quantity for each Date. A NUMBER\_QUANTITY Calendar dictates the values during the horizon', and the before\_horizon\_number', after\_horizon\_number', before\_horizon\_quantity', and after\_horizon\_quantity' fields specify the values used beyond the horizon'.

Note that currently only unitless Quantities are supported. In the future, the QUANTITY Calendar will have a 'unit' field which defines the Unit to which all the entries' quantities are converted.

before\_horizon\_number -- a Number field of model NUMBER\_QUANTITY

This will be the value of this Calendar\_Plan at any Date before the horizon'.

This is not settable. It is dictated by the model that provides the Calendar\_Plan field. See the documentation for the Calendar\_Plan field of interest.

Properties: Export-Only Field

after\_horizon\_number -- a Number field of model NUMBER\_QUANTITY

This will be the value of this Calendar\_Plan at any Date after the horizon'.

This is not settable. It is dictated by the model that provides the Calendar\_Plan field. See the documentation for the Calendar\_Plan field of interest.

Properties: Export-Only Field

<i>Extensions</i>	<i>SYMBOL Extension</i>
<p><b>intra_horizon_number</b> - - <i>a Number field of model NUMBER_QUANTITY</i>  This will be the value of this Calendar_Plan at any Date within the 'horizon' where no other Calendar_Entry is active. If not set, the default from 'calendar' is used.  Default: 0</p> <p><b>before_horizon_quantity</b> - - <i>a Quantity field of model NUMBER_QUANTITY</i>  This will be the value of this Calendar_Plan at any Date before the 'horizon'.</p> <p>This is not settable. It is dictated by the model that provides the Calendar_Plan field.  See the documentation for the Calendar_Plan field of interest.  Properties: Export-Only Field</p> <p><b>after_horizon_quantity</b> - - <i>a Quantity field of model NUMBER_QUANTITY</i>  This will be the value of this Calendar_Plan at any Date after the 'horizon'.</p> <p>This is not settable. It is dictated by the model that provides the Calendar_Plan field.  See the documentation for the Calendar_Plan field of interest.  Properties: Export-Only Field</p> <p><b>intra_horizon_quantity</b> - - <i>a Quantity field of model NUMBER_QUANTITY</i>  This will be the value of this Calendar_Plan at any Date within the 'horizon' where no other Calendar_Entry is active. If not set, the default from 'calendar' is used.  Default: 0 (unitless)</p> <p><b>SYMBOL</b> - - <i>a entry_value extension of model Calendar_Plan</i></p> <p>A SYMBOL Calendar_Plan contains a Symbol for each Date. A SYMBOL Calendar dictates the values during the 'horizon', and the 'before_horizon_symbol' and 'after_horizon_symbol' fields specify the values used beyond the 'horizon'.</p> <p><b>before_horizon_symbol</b> - - <i>a Symbol field of model SYMBOL</i>  This will be the value of this Calendar_Plan at any Date before the 'horizon'.</p> <p>This is not settable. It is dictated by the model that provides the Calendar_Plan field.  See the documentation for the Calendar_Plan field of interest.  Properties: Export-Only Field</p> <p><b>after_horizon_symbol</b> - - <i>a Symbol field of model SYMBOL</i>  This will be the value of this Calendar_Plan at any Date after the 'horizon'.</p> <p>This is not settable. It is dictated by the model that provides the Calendar_Plan field.  See the documentation for the Calendar_Plan field of interest.</p>	

<i>Extensions</i>	<i>TIME Extension</i>
<p>Properties: Export-Only Field</p> <p><b>intra_horizon_symbol</b> - - <i>a Symbol field of model SYMBOL</i>  This will be the value of this Calendar_Plan at any Date within the 'horizon' where no other Calendar_Entry is active. If not set, the default from 'calendar' is used.  Default: none</p> <p><b>TIME</b> - - <i>a entry_value extension of model Calendar_Plan</i></p> <p>A TIME Calendar_Plan contains a Time for each Date. A TIME Calendar dictates the values during the 'horizon', and the 'before_horizon_time' and 'after_horizon_time' fields specify the values used beyond the 'horizon'.</p> <p><b>before_horizon_time</b> - - <i>a Time field of model TIME</i>  This will be the value of this Calendar_Plan at any Date before the 'horizon'.</p> <p>This is not settable. It is dictated by the model that provides the Calendar_Plan field.  See the documentation for the Calendar_Plan field of interest.  Properties: Export-Only Field</p> <p><b>after_horizon_time</b> - - <i>a Time field of model TIME</i>  This will be the value of this Calendar_Plan at any Date after the 'horizon'.</p> <p>This is not settable. It is dictated by the model that provides the Calendar_Plan field.  See the documentation for the Calendar_Plan field of interest.  Properties: Export-Only Field</p> <p><b>intra_horizon_time</b> - - <i>a Time field of model TIME</i>  This will be the value of this Calendar_Plan at any Date within the 'horizon' where no other Calendar_Entry is active. If not set, the default from 'calendar' is used.  Default: 00:00</p>	

Extensions	Format Extensions
------------	-------------------

### 10.45 Format Extensions

#### 10.45.1 spec extensions of model Format

Void -- a spec extension of model Format

A Format which can handle Void values (or lack of such values).

These models have a field that is a Void model :

Plan, Operation\_Plan, Resource\_Plan, Forecast, Seller\_Plan, Problem, Active\_Strategy, LINK, Operation\_State, Request, Promise, Acceptance, Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance, Strategy, Forecast\_Entry, Item\_Request, Load\_Plan, Item\_Acceptance, Delivery\_Available\_To\_Promise, Item\_Promise, Item\_Available\_To\_Promise, Product\_Available\_To\_Promise, MEMBER\_RANK, ALTERNATES\_PRIMARY, ALTERNATES\_PROPORTIONAL, EFFECTIVE\_CALENDAR, Consolidation, BUCKETED\_NESTED\_SORT, User, Data\_Directory.

specification -- a Symbol field of model Void

Specifies how to format void values

Default: Empty String

Logical -- a spec extension of model Format

A Format which can handle Logical values.

These models have a field that is a Logical model :

Site, Seller, Operation, Operation\_Plan, Product, Forecast, Problem, Active\_Strategy, LINK, Location, Item, Configuration, Item\_Group, CUSTOMER, Product\_Group, Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance, Strategy, Problem\_Set, Active\_Problem, MANUAL, Unit, Forecast\_Entry, GROUP, Item\_Request, Flow, Operation\_Problem\_Detector, Load\_Plan, Flow\_Plan, Item\_Acceptance, Item\_Promise, Buffer\_Problem\_Detector, Lot, Box, DELIVER, Resource\_Problem\_Detector, MEMBER\_RANK, ALTERNATES\_PRIMARY, ALTERNATES\_MANUAL, ALTERNATES\_PROPORTIONAL, ALTERNATES\_PREFERENCE, EFFECTIVE\_CALENDAR, MPPS\_Operation\_Resource, MPPS\_BASIC, CALENDAR\_RATE, BUCKETED\_NESTED\_SORT, Sorted\_Bucket, PRODUCING\_FLOW\_CALENDAR, PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK,

Extensions	String Extension
------------	------------------

CUSTOMER\_RANK, SELLER\_RANK, REQUEST\_RANK, PROMISE\_RANK, ACTUAL\_OR\_FORECAST, REQUEST\_ISSUED, PROMISE\_DUE, REQUEST\_DUE, DUE\_DATE, PROMISED, ENTRY\_DATE, Feature, User, Data\_Directory, Report\_Directory, Format, Model\_Type, Field.

specification -- a Symbol field of model Logical

Specifies how to format Logical values

Default: False, True

String -- a spec extension of model Format

A Format which can handle Strings.

These models have a field that is a String model :

Supply\_Chain, Site, Seller, Plan, Problem\_Category, Operation, Operation\_Plan, Resource, Resource\_Plan, Buffer, Buffer\_Plan, Product\_Root, Product, Forecast, Site\_Plan, Active\_Strategy, Location, Item, Skill, Item\_Group, Request, Horizon, Site\_Group, Product\_Group, Delivery\_Request, Strategy, Lot, Calendar\_Entry, Calendar, Option, DAYS\_OF\_WEEK, DAY\_OF\_MONTH, DAY\_OF\_WEEK\_OF\_MONTH, DAY\_OF\_LAST\_WEEK\_OF\_MONTH, User, Data\_Directory, Report\_Directory, Format, Date\_Range, Quantity\_Range, Horizon\_Date, List, Model\_Type, Field, Extension\_Selector, Field\_Error.

specification -- a Symbol field of model String

Specifies how to format Strings. A "\\$" in the format string specifies the place at which the string should be inserted.

Default: "\$"

Symbol -- a spec extension of model Format

A Format which can handle Strings.

These models have a field that is a Symbol model :

Supply\_Chain, Site, Seller, Plan, Problem\_Category, Operation, Operation\_Plan, Resource, Resource\_Plan, Buffer, Product\_Root, Product, Forecast, Location, Item, Skill, Item\_Group, Request, Horizon, Site\_Group, Product\_Group, Delivery\_Request, Strategy, Strategy\_Lock, Item\_Request, Load, Flow, Lot, Load\_Size, SETUP, SKILL, Calendar\_Entry, Calendar, SYMBOL, ACCESSORY, OPTION,

Extensions	Date Extension
------------	----------------

INCOMPATIBLE\_ACCESSORY, INCOMPATIBLE\_OPTION, EARLIEST, MPPS\_Operation\_Resource, REQUEST\_FIXED, REQUEST\_FIXED\_WITH\_ANALYSIS, Time\_Table\_Entry, Consolidation, CONSOLIDATION\_OVERSIZE, CONSOLIDATION\_UNDERSIZE, BLOCK\_CYCLE, MPPS\_Batch, Size\_Dimension, Feature, Option, Option\_Spec, CALENDAR, SYMBOL, SYMBOL, User, Format, Void, Logical, String, Symbol, Date, Date\_Range, Number, Percentage, Integer, Quantity, Quantity\_Range, Time, Restriction, List, Model\_Type, Field, Extension\_Selector.

specification - - a Symbol field of model Symbol  
 Specifies how to format Symbols. A "\\$" in the format string specifies the place at which the string should be inserted.  
 Default: "\$"

Date -- a spec extension of model Format

A Format which can handle Dates input or output in a variety of ways.

These models have a field that is a Date model :  
 Plan, Operation, Operation\_Plan, Resource\_Plan, Buffer\_Plan, Problem, Active\_Strategy, Operation\_State, Request, Promise, Acceptance, Horizon\_Bucket\_Start, Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance, Forecast\_Entry, Item\_Request, Item\_Acceptance, Delivery\_Available\_To\_Promise, Item\_Promise, Profile\_Number, Profile\_Percentage, Profile\_Quantity, Yield\_Changes, END, HOLD, Efficiency\_Change, RAMP\_LINEAR, REQUEST\_ISSUED, PROMISE\_DUE, REQUEST\_DUE, DUE\_DATE.

specification - - a Symbol field of model Date  
 Specifies how to format Date values. Special formatting characters are substituted with the corresponding date elements. All other characters will be part of the resulting date string.

The special formatting characters:

ll == Hour in 12 hour format, leading zero

\_l == Hour in 12 hour format, no leading zero

hh == Hour in 24 hour format, leading zero

Extensions	Date Extension
------------	----------------

\_h == Hour in 24 hour format, no leading zero

mm == Minute, leading zero

\_m == Minute, no leading zero

ss == Second, leading zero

\_s == Second, no leading zero

AP == AM/PM flag, uppercase

ap == AM/PM flag, lowercase

zzz == date zone abbreviation

DD == Day of month, leading zero

\_D == Day of month, no leading zero

MM == Month of year, leading zero

\_M == Month of year, no leading zero

MMM == Month Abbreviation, all caps

Mmm == Month Abbreviation, capitalized

mmm == Month Abbreviation, lowercase

MR == Rounded Month of year, leading zero (month rounded up when day > 21)

\_R == Rounded Month of year, no leading zero (month rounded up when day > 21)

MMR == Rounded Month Abbreviation, all caps

Mmr == Rounded Month Abbreviation, capitalized

mmr == Rounded Month Abbreviation, lowercase

YY == Year mod 100

Extensions	Date_Range Extension
------------	----------------------

YYYY == Year

WW == 2 character Day of week abbreviation, all caps

Ww == 2 character Day of week abbreviation, Capitalized

ww == 2 character Day of week abbreviation, lowercase

WWW == 3 character Day of week abbreviation, all caps

Www == 3 character Day of week abbreviation, Capitalized

www == 3 character Day of week abbreviation, lowercase

Default: MM/DD/YY hh:mm:ss

Date\_Range -- a spec extension of model Format

A Format which can handle Date\_Ranges

These models have a field that is a Date\_Range model :  
Plan, Operation\_Plan, Product\_Root, Problem, Delivery\_Request,  
Delivery\_Promise, Delivery\_Acceptance, Forecast\_Entry, ATP\_Entry,  
Load\_Plan, Flow\_Plan, Item\_Available\_To\_Promise,  
Product\_Available\_To\_Promise, Calendar\_Entry, Sorted\_Bucket,  
Calendar\_Plan.

specification -- a Symbol field of model Date\_Range  
Specifies how to format Date\_Range values  
Default: MM/DD/YY hh:mm:ss

separator -- a String field of model Date\_Range  
Specifies the separator between the min & max dates.  
Default: /

Number -- a spec extension of model Format

A Format which can handle Integers and Numbers

These models have a field that is a Number model :

Extensions	Number Extension
------------	------------------

Site, Seller, Operation\_Plan, Product, Problem, Active\_Strategy,  
WEEKS, DAYS\_WEEKS, DAYS, Delivery\_Request, Delivery\_Promise,  
Strategy, OPERATION\_PLAN\_RANK\_RANGE, FEASIBILITY,  
MINIMIZE\_PROBLEM\_COUNT, MINIMIZE\_PROBLEMS,  
MINIMIZE\_LATENESS, WEIGHTED\_LATENESS,  
MINIMIZE\_SHORTNESS, WEIGHTED\_SHORTNESS,  
MINIMIZE\_COST, MAXIMIZE\_PROFIT, MAXIMIZE\_REVENUE,  
Active\_Problem, Active\_Goal, FEASIBILITY,  
MINIMIZE\_PROBLEM\_COUNT, MINIMIZE\_PROBLEMS,  
MINIMIZE\_LATENESS, WEIGHTED\_LATENESS,  
WEIGHTED\_SHORTNESS, MINIMIZE\_SHORTNESS,  
MINIMIZE\_COST, MAXIMIZE\_PROFIT, MAXIMIZE\_REVENUE,  
Alternate\_Product, Profile\_Number, Ordered\_Sub\_Strategy,  
SEQUENCE\_RUN\_ONCE, SEQUENCE\_RUN\_MULTIPLE, Box,  
Calendar\_Entry, NUMBER, NUMBER\_QUANTITY, MAX, MIN, MAX,  
TIME\_MULTIPLE, MPPS\_Operation\_Resource, Routing\_Operation,  
MAX\_SETUPS\_PERIOD, Flow\_Criterion, CUSTOMER\_RANK,  
SELLER\_RANK, REQUEST\_RANK, PROMISE\_RANK,  
ACTUAL\_OR\_FORECAST, REQUEST\_ISSUED, PROMISE\_DUE,  
REQUEST\_DUE, DUE\_DATE, PROMISED, ENTRY\_DATE, NUMBER,  
NUMBER\_QUANTITY, NUMBER, NUMBER\_QUANTITY,  
Data\_Directory, Report\_Directory.

specification -- a Symbol field of model Number

1) A number format is specified using '0', '#' and '?'. A '0' puts that digit or '0' if there is no number at that position. A '?' puts the number at that position or a ' ' (space) if there is no number at that position. A '#' puts the number at that position if there is a number, else it does not put anything. A decimal point ('.') separates the decimal part from the fractional part. The total number of zero + pound + question marks to the right of the point denotes the maximum number of digits to be displayed after the point. There can be characters within the decimal part of the number. 2) Anything within double quotes are copied directly to the output. 3) Single characters '\$', ',', '+', '-', '(', ')', '\*' in the format are copied directly to the output. 4) Any character after a 'h' is displayed only if the number is negative. Otherwise a space is displayed there. If there is no 'h' in the format, negative numbers will be displayed without any '-' sign. See Example (4), 5) Any character after a backslash('\') is displayed as it is. 6) A comma at any place in the format means that the number should be displayed in comma notation. 7) A number of blanks may be specified using Sxx, where xx is the 2 digit number of blanks. Note that xx \* must\* be 2 digits. This feature may be used to provide room in the string for the units to be added later. Example 2 specifies an 8 character buffer to follow the number.

Examples: -----format\_ = n(##n) exp\_format\_(compiled format\_) =  
(#####) Input number: 2.34234e+08 is displayed as 234234312.23  
Input number: -1.23423e+06 is displayed as (1234234.1) Input number: 1.23423e+06  
is displayed as 1234234.12 Input number: 1.23442e+07 is displayed as  
12344234.13

===== format\_ = n-#0#S08 exp\_format\_(compiled format\_) = -

##### Input number: 2.34234e+08 is displayed as 234234312.23  
Input number: -1.23423e+06 is displayed as -1234234.1 Input number: 1.23423e+06  
is displayed as 1234234.12 Input number: 1.23442e+07 is displayed as 12344234.13

===== format\_ = n(##0.0#n)"CR" exp\_format\_(compiled format\_) =

(#####) CR Input number: 2.34234e+08 is displayed as  
234.234.312.23 CR Input number: -1.23423e+06 is displayed as (1.234.234.1) CR  
Input number: 1.23423e+06 is displayed as 1.234.234.12 CR Input number:  
1.23442e+07 is displayed as 12.344.234.13 CR

===== format\_ = "X="0.0 exp\_format\_(compiled format\_) =

X=##### Input number: 2.34234e+08 is displayed as X=234234312.2  
Input number: -1.23423e+06 is displayed as X=-1234234.1 Input number:  
1.23423e+06 is displayed as X=1234234.1 Input number:  
1.23442e+07 is displayed as X=12344234.1

===== format\_ = \$(n(???00.0?n) exp\_format\_(compiled format\_) = \$(#####

00.0 ) Input number: 2.34234e+08 is displayed as \$ 234234312.234 Input number: -  
1.23423e+06 is displayed as \$(1234234.1 ) Input number:  
1.23423e+06 is displayed as \$ 1234234.123 Input number:  
1.23442e+07 is displayed as \$ 12344234.135

===== format\_ = \$n-?.??? exp\_format\_(compiled format\_) = \$-#.#.#.#.#

01 Input number: 2.34234e+08 is displayed as \$ 234.234.3121 Input number: -  
1.23423e+06 is displayed as \$-1.234.2341 Input number:  
1.23423e+06 is displayed as \$ 1.234.2341 Input number: 34.1345 is  
displayed as \$ 341  
Default: n-#.#.#

Percentage -- a spec extension of model Format

A Format which can handle Percentages.

These models have a field that is a Percentage model:

Operation\_Plan, Resource\_Plan, Delivery\_Promise, Strategy,  
Problem\_Set, Strategy\_Change, Strategy\_Goal, Active\_Problem,  
Sub\_Product\_Group, Sub\_Product, Item\_Promise, Profile\_Percentage,  
Ranked\_Sub\_Strategy, MAINTENANCE, Resource\_Skill,  
Skill\_Resource, PER\_ALLOCATED, PER\_COMMITTED,  
MEMBER\_RANK, FIXED\_SPLIT, Product\_Allocation,  
Alternate\_Operation, Effective\_Calendar\_Operation, PRODUCE\_YIELD,  
PRODUCE\_YIELD\_CALENDAR, Yield\_Changes,  
PRODUCE\_YIELD\_RAMP\_CALENDAR, MAX, MIN\_MAX, MPPS,  
MPPS\_Operation\_Resource, REWORK, Consolidation,  
CONSOLIDATION\_OVERSIZE, CONSOLIDATION\_OVERSIZE,  
CONSOLIDATION\_UNDERSIZE, CONSOLIDATION\_UNDERSIZE,  
FIXED, Efficiency\_Change, RAMP\_LINEAR, SHARED\_USE, OVER-  
SIZE, OVERSIZE, BUCKET\_OVERSIZE, BUCKET\_OVERSIZE,  
UNDERLOAD, UNDERLOAD, CUMULATIVE\_OVERLOAD,  
CUMULATIVE\_OVERLOAD, FIXED, FIXED,  
BUCKETED\_NESTED\_SORT, BUCKETED\_COMBINED\_SORT,  
Option\_Spec, CALENDAR.

specification - - a Symbol field of model Percentage  
Exactly the same as Number format specifications. Example: specification = "%#%"  
formats 23 into "23%"  
Default: n-#.#"%"

Integer -- a spec extension of model Format

A Format which can handle Integers.

These models have a field that is a Integer model:

Site, Seller, Operation, Operation\_Plan, Resource, Resource\_Plan,  
Buffer, Buffer\_Plan, Product\_Root, Product, Forecast, Site\_Plan, Prob-  
lem, Active\_Strategy, LINK, Item, Skill, Configuration, SUPPLIER,  
CUSTOMER, LINK, Operation\_State, Request, Promise, SUPPLIER,  
CUSTOMER, Horizon, ONE, MONTHS, WEEKS, DAYS, WEEKS,  
DAYS, LIST\_HORIZON\_DATES, DATES,  
MONTHS\_WITH\_FULL\_WEEKS, Site\_Group, Delivery\_Request,  
Delivery\_Promise, Delivery\_Acceptance, Strategy, Problem\_Set,  
Strategy\_Change, Strategy\_Lock, Strategy\_Goal,  
OPERATION\_PLAN\_RANK\_RANGE,  
OPERATION\_PLAN\_RANK\_EXPRESSION, Active\_Goal,  
Forecast\_Entry, Load, Flow, Operation\_Problem\_Detector,  
Buffer\_Problem\_Detector, USE\_ALTERNATE\_OPERATION,



*Extensions**Integer Extension*

REQUEST\_NOT\_PLANNED, REQUEST\_PLANNED\_LATE,  
 REQUEST\_PLANNED\_EARLY, REQUEST\_PLANNED\_SHORT,  
 REQUEST\_PLANNED\_EXCESS, PROMISE\_NOT\_PLANNED,  
 PROMISE\_PLANNED\_LATE, PROMISE\_PLANNED\_EARLY,  
 PROMISE\_PLANNED\_SHORT, PROMISE\_PLANNED\_EXCESS,  
 ACCEPTANCE\_NOT\_PLANNED, ACCEPTANCE\_PLANNED\_LATE,  
 ACCEPTANCE\_PLANNED\_EARLY, ACCEPTANCE\_PLANNED\_SHORT,  
 ACCEPTANCE\_PLANNED\_EXCESS, SEQUENCE\_RUN\_ONCE,  
 SEQUENCE\_RUN\_MULTIPLE, SEQUENTIAL\_ALTERNATES,  
 SEQUENTIAL\_ALTERNATES, SEQUENTIAL\_ALTERNATES\_KEEP\_BEST,  
 SEQUENTIAL\_ALTERNATES\_KEEP\_BEST,  
 PROPORTIONAL\_RESOLVES, ORDERED\_RESOLVES, Load\_Size,  
 PRECEDENCE, OVER\_RESTRICTION, EXPEDITED,  
 PLANNED\_BEFORE\_CURRENT, UNRELEASED, NEEDS\_RELEASE,  
 INCONSISTENT\_OPLAN, OPERATION, REQUEST\_PROMISED\_LATE,  
 REQUEST\_PROMISED\_EARLY, REQUEST\_PROMISED\_SHORT,  
 REQUEST\_PROMISED\_EXCESS, PROMISE\_NOT\_OFFERED,  
 PROMISE\_NOT\_CONFIRMED, PROMISE\_NOT\_ACCEPTED,  
 ACCEPTANCE\_INCONSISTENT, REQUEST\_QUEUED, REQUEST,  
 REQUEST\_PLAN, PROMISE, PROMISE\_PLAN, REQUEST\_PROMISE,  
 DELIVERY\_REQUEST\_NOT\_COORDINATED,  
 DELIVERY\_REQUEST\_NOT\_COORDINATED,  
 DELIVERY\_PROMISE\_NOT\_COORDINATED,  
 DELIVERY\_ACCEPTANCE\_NOT\_COORDINATED, Resource\_Skill,  
 Resource\_Problem\_Detector, Skill\_Resource,  
 PROPORTIONAL\_INTERACTION, EARLIEST\_PROBLEM\_START,  
 SORT\_BY\_EXPRESSION, SUPPLY\_PLANNED\_LATE,  
 SUPPLY\_PLANNED\_EARLY, SUPPLY\_PLANNED\_SHORT,  
 SUPPLY\_PLANNED\_EXCESS, SUPPLY\_PROMISED\_LATE,  
 SUPPLY\_PROMISED\_EARLY, SUPPLY\_PROMISED\_SHORT,  
 SUPPLY\_PROMISED\_EXCESS, SUPPLY, SUPPLY\_PLAN,  
 SUPPLY\_PROMISE, FCFS, PER\_ALLOCATED, PER\_COMMITTED,  
 MEMBER\_RANK, FIXED\_SPLIT, CALENDAR\_SPLIT, SLIDING,  
 HORIZON, BUCKETED\_ALL, BUCKETED\_ASAP, FIXED, FIXED,  
 AT\_END, SIMPLE\_FIXED\_QUANTITY, SINGLE\_REQUEST,  
 SIMPLE\_FIXED\_TIME, WEEKLY, DUAL\_REQUEST, ON\_TIME,  
 ON\_TIME, ON\_TIME, FULL\_QUANTITIES\_OF\_ALL\_ITEMS,  
 FULL\_QUANTITIES\_OF\_ALL\_ITEMS,  
 FULL\_QUANTITIES\_OF\_ALL\_ITEMS, UNRESTRICTED, UNRE-  
 STRICTED, UNRESTRICTED, NUMBERED, NONE, NONE, FIXED,  
 ON\_TIME, ALL, ALL\_ON\_TIME, ASAP, ASAP\_MONTHLY,  
 BUCKETED\_ALLOCATION, BUCKETED\_ALL\_MIN\_PRICE,

*Extensions**Integer Extension*

BUCKETED\_MIN\_PRICE\_ASAP, SHIP\_IN\_RATIO, Calendar\_Entry,  
 Calendar, UNSPECIFIED, NUMBER, QUANTITY,  
 NUMBER\_QUANTITY, SYMBOL, TIME, ALTERNATES\_PRIMARY,  
 Alternate\_Operation, ALTERNATES\_PROPORTIONAL,  
 EFFECTIVE\_CALENDAR, CONSUME\_FIXED, CONSUME\_PER,  
 PRODUCE\_FIXED, PRODUCE\_PER, PRODUCE\_YIELD,  
 PRODUCE\_YIELD\_CALENDAR, PRODUCE\_YIELD\_RAMP\_LIST,  
 PRODUCE\_YIELD\_RAMP\_CALENDAR, EARLIEST\_FIXED, LINEAR,  
 RESOURCE, ONE, MAX, MIN\_MAX, UNIDENTIFIED\_OP\_STATE,  
 DELAY\_ONLY\_FIXED, DELAY\_ONLY\_BASIC,  
 DELAY\_ONLY\_CALENDAR, BASIC\_CALENDARS, FIXED\_TIME,  
 TIME\_MULTIPLE, BASIC, BASIC\_DELAYED, FILL, OFFSET,  
 TIME\_EXPRESSION, REQUEST\_FIXED,  
 REQUEST\_FIXED\_WITH\_ANALYSIS, ROUTING, STARTED, COM-  
 PLETED, IN\_FRONT, SIMPLE\_CONSOLIDATION, UNCONSOLI-  
 DATED, UNCOORDINATED, CONSOLIDATION\_OVERSIZE,  
 CONSOLIDATION\_UNDERSIZE, FIXED\_STEP\_LIST, RAMP\_LINEAR,  
 RAMP\_LIST, CALENDAR, RAMP\_CALENDAR, LOAD\_TIME,  
 WEIGHTED\_TIME, INFINITE\_USE, EXCLUSIVE\_USE, SHARED\_USE,  
 ZERO, OVERLOAD, OVERTIME, OVERSIZE, BUCKET\_OVERSIZE,  
 UNDERLOAD, CUMULATIVE\_OVERLOAD, RESOURCE, FIXED,  
 CALENDAR, PRIMARY, PREFER\_PRIMARY, EVEN,  
 MAX\_EFFICIENCY, UNLIMITED, FIXED\_COUNT,  
 FIXED\_QUANTITY, CALENDAR\_COUNT, CALENDAR\_QUANTITY,  
 MULTI\_DIMENSION, CALENDAR\_RATE, FIXED, CALENDAR,  
 ZERO, FIXED, NEGATIVE\_ON\_HAND, OVER\_FLOW\_LIMIT,  
 NEGATIVE\_ON\_HAND\_AT\_END, LOT\_OVER\_CONSUMED,  
 LOT\_NOT\_CONSUMED, LOT\_NOT\_PRODUCED,  
 LOT\_OVER\_PRODUCED, LOW\_ON\_HAND, EXCESS\_ON\_HAND,  
 EXCESS\_ON\_HAND\_AT\_END, BUFFER, BUCKETED\_NESTED\_SORT,  
 Flow\_Criterion, BUCKETED\_COMBINED\_SORT,  
 PRODUCING\_FLOW\_CALENDAR,  
 PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK,  
 SUPPLY\_CALENDAR, ON\_HAND\_CALENDAR,  
 ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK,  
 FLOW\_LIMIT\_CALENDAR,  
 FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK, CUSTOMER\_RANK,  
 SELLER\_RANK, REQUEST\_RANK, ACTUAL\_OR\_FORECAST,  
 REQUEST\_ISSUED, PROMISE\_DUE, REQUEST\_DUE, DUE\_DATE,  
 PROMISED, ENTRY\_DATE, INFINITE, SUPPLIER, FLOW\_THRU,  
 PHANTOM, BASIC, BASIC\_FILTER\_AND\_RANK, FIXED\_QUANTITY,  
 MULTIPLE, MULTIPLE\_FILTER\_AND\_RANK,

Extensions	Quantity Extension
------------	--------------------

FIXED\_QUANTITY\_FENCED, FIXED\_TIME, STANDARD, CUSTOM, LFL\_SIMPLE, LFL\_BOUNDED, MLFL\_BOUNDED, MANUAL, CALENDAR, Calendar\_Plan, EVERYDAY, EVERY\_N\_DAYS, WEEKDAYS, WEEKENDS, DAYS\_OF\_WEEK, DAY\_OF\_MONTH, DAY\_OF\_WEEK\_OF\_MONTH, DAY\_OF\_LAST\_WEEK\_OF\_MONTH, DAY\_OF\_YEAR, YEARLY, Format, Void, Logical, String, Symbol, Date, Date\_Range, Number, Percentage, Integer, Quantity, Quantity\_Range, Time, Restriction, Horizon\_Date, List, Field.

specification -- a Symbol field of model Integer  
 Specifies how to format Integers. See NUMBER specification.  
 Default: %d

Quantity -- a spec extension of model Format

A Format which can handle Quantities.

These models have a field that is a Quantity model :

Operation\_Plan, Resource\_Plan, Buffer\_Plan, Product\_Root, Product, Location, LINK, Strategy, Unit, Unit\_Quantity, Generic\_Product, Alternate\_Product, Sub\_Product\_Group, Sub\_Product, Forecast\_Entry, ATP\_Entry, Item\_Request, Flow, Load\_Plan, Flow\_Plan, Item\_Acceptance, Item\_Promise, Item\_Available\_To\_Promise, Product\_Available\_To\_Promise, Lot\_Flow, Profile\_Number, Profile\_Percentage, Profile\_Quantity, Lot, REQUEST\_PLANNED\_SHORT, REQUEST\_PLANNED\_EXCESS, PROMISE\_PLANNED\_SHORT, PROMISE\_PLANNED\_EXCESS, ACCEPTANCE\_PLANNED\_SHORT, ACCEPTANCE\_PLANNED\_EXCESS, REQUEST\_PROMISED\_SHORT, REQUEST\_PROMISED\_EXCESS, SUPPLY\_PLANNED\_SHORT, SUPPLY\_PLANNED\_EXCESS, SUPPLY\_PROMISED\_SHORT, SUPPLY\_PROMISED\_EXCESS, MEMBER\_RANK, SIMPLE\_FIXED\_QUANTITY, SINGLE\_REQUEST, SIMPLE\_FIXED\_TIME, WEEKLY, DUAL\_REQUEST, Calendar\_Entry, QUANTITY, NUMBER\_QUANTITY, Alternate\_Operation, Effective\_Calendar\_Operation, CONSUME\_FIXED, CONSUME\_PER, PRODUCE\_FIXED, PRODUCE\_PER, PRODUCE\_YIELD, PRODUCE\_YIELD\_CALENDAR, PRODUCE\_YIELD\_RAMP\_LIST, PRODUCE\_YIELD\_RAMP\_CALENDAR, FIXED, LINEAR, Routing\_Operation, REMAINING\_TIME, COMPLETED\_TIME, PERCENT\_COMPLETE, CUMULATIVE, ARRIVED, STARTED, SCRAPPED, COMPLETED, ON\_HAND, IN\_FRONT, IN\_PROCESS, IN\_BACK, Unordered\_Operation,

Extensions	Quantity_Range Extension
------------	--------------------------

MPPS\_BATCHING, FIXED\_QUANTITY, Size\_Dimension, MIN\_ON\_HAND, MAX\_ON\_HAND, NEGATIVE\_ON\_HAND, NEGATIVE\_ON\_HAND, OVER\_FLOW\_LIMIT, OVER\_FLOW\_LIMIT, NEGATIVE\_ON\_HAND\_AT\_END, NEGATIVE\_ON\_HAND\_AT\_END, LOT\_OVER\_CONSUMED, LOT\_OVER\_PRODUCED, LOW\_ON\_HAND, LOW\_ON\_HAND, EXCESS\_ON\_HAND, EXCESS\_ON\_HAND, EXCESS\_ON\_HAND\_AT\_END, BUCKETED\_NESTED\_SORT, EXCESS\_ON\_HAND\_AT\_END, BUCKETED\_NESTED\_SORT, Sorted\_Bucket, BUCKETED\_COMBINED\_SORT, CALENDAR, PRODUCING\_FLOW\_CALENDAR, PRODUCING\_FLOW\_CALENDAR, PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK, PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK, ON\_HAND\_CALENDAR, ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK, FLOW\_LIMIT\_CALENDAR, FLOW\_LIMIT\_CALENDAR\_FILTER\_AND\_RANK, FLOW\_MIN\_TIME, FLOW\_MIN\_ON\_HAND, BASIC, BASIC, BASIC\_FILTER\_AND\_RANK, BASIC\_FILTER\_AND\_RANK, FIXED\_QUANTITY, FIXED\_QUANTITY, MULTIPLE, MULTIPLE, MULTIPLE\_FILTER\_AND\_RANK, MULTIPLE\_FILTER\_AND\_RANK, FIXED\_QUANTITY\_FENCED, FIXED\_QUANTITY\_FENCED, FIXED\_TIME, FIXED\_TIME, CALENDAR, QUANTITY, NUMBER\_QUANTITY, QUANTITY, NUMBER\_QUANTITY.

specification -- a Symbol field of model Quantity  
 Specifies how to format Quantities. This is a combination of a NUMBER specification and a Measure\_Unit specification.  
 Default: n-##.## S3

Quantity\_Range -- a spec extension of model Format

A Format which can handle Quantity Ranges.

These models have a field that is a Quantity\_Range model :

Item\_Request, Item\_Acceptance, Item\_Promise, PRODUCING\_FLOW\_CALENDAR, PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK, MULTIPLE, MULTIPLE\_FILTER\_AND\_RANK, LFL\_BOUNDED, MLFL\_BOUNDED.

specification -- a Symbol field of model Quantity\_Range  
 Specifies how to format Quantity\_Range values  
 Default: n-##.## S3

Extensions	Time Extension
------------	----------------

separator . . . a String field of model Quantity\_Range  
 Specifies the separator between the min & max quantities.  
 Default: <

Time -- a spec extension of model Format

A Format which can handle Times.

These models have a field that is a Time model :

Operation\_Plan, Resource\_Plan, Product\_Root, Product, Active\_Strategy,  
 Delivery\_Request, Delivery\_Promise, Delivery\_Acceptance, Strategy,  
 Problem\_Set, Generic\_Product, Alternate\_Product, Item\_Request,  
 Item\_Acceptance, Item\_Promise, REQUEST\_PLANNED\_LATE,  
 REQUEST\_PLANNED\_EARLY, PROMISE\_PLANNED\_LATE,  
 PROMISE\_PLANNED\_EARLY, ACCEPTANCE\_PLANNED\_LATE,  
 ACCEPTANCE\_PLANNED\_EARLY, PRECEDENCE,  
 REQUEST\_PROMISED\_LATE, REQUEST\_PROMISED\_EARLY,  
 SUPPLY\_PLANNED\_LATE, SUPPLY\_PLANNED\_EARLY,  
 SUPPLY\_PROMISED\_LATE, SUPPLY\_PROMISED\_EARLY,  
 SIMPLE\_FIXED\_TIME, Calendar\_Entry, TIME, PRODUCE\_YIELD,  
 PRODUCE\_YIELD\_CALENDAR, PRODUCE\_YIELD\_RAMP\_LIST,  
 PRODUCE\_YIELD\_RAMP\_CALENDAR, DELAY\_ONLY\_FIXED,  
 DELAY\_ONLY\_BASIC, FIXED\_TIME, TIME\_MULTIPLE, BASIC,  
 BASIC\_DELAYED, MPPS, REQUEST\_FIXED,  
 REQUEST\_FIXED\_WITH\_ANALYSIS, Request\_Alternates,  
 SETUP\_FIXED, SETUP\_TABLE, Time\_Table\_Entry,  
 SETUP\_CALENDAR, SETUP\_EXPRESSION, SKILL\_FIXED,  
 SKILL\_TABLE, SKILL\_CALENDAR, SKILL\_EXPRESSION,  
 TRANSIT\_FIXED, TRANSIT\_TABLE, TRANSIT\_CALENDAR,  
 TRANSIT\_EXPRESSION, SHARED\_USE, MPPS\_BASIC, LOAD\_TIME,  
 DUAL\_WEIGHTED\_TIME, MAX\_SETUPS\_PERIOD,  
 MAX\_SETUP\_TIME\_PER\_PERIOD, OVERLOAD, OVERLOAD,  
 TIME, OVERTIME, FIXED, MTBF, LOT\_OVER\_CONSUMED,  
 LOT\_OVER\_PRODUCED, LOT\_OVER\_PRODUCED,  
 PRODUCING\_FLOW\_CALENDAR,  
 PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK,  
 FLOW\_MIN\_TIME, BASIC, BASIC\_FILTER\_AND\_RANK,  
 FIXED\_QUANTITY, MULTIPLE, MULTIPLE\_FILTER\_AND\_RANK,  
 FIXED\_QUANTITY\_FENCED, FIXED\_TIME, LFL\_MIN\_TIME, CAL-  
 ENDAR, TIME, TIME.

Extensions	Restriction Extension
------------	-----------------------

specification . . . a Symbol field of model Time  
 Specifies how to format Time values. This can be as a Quantity (see QUANTITY) or  
 in hour-minute-second form. The letter 'h' is substituted for hours, the letter 'm' for  
 minutes, and the letter 's' for seconds (similar to the DATE Format).  
 Default: hh:mm

Restriction -- a spec extension of model Format

A Format which can handle Restriction objects (see the documentation for the restric-  
 tion type). The format is:

```

input      period start end  within latest  output
=====
unspecified  (oo,oo) f  f  f  f  f  [unspecified]

start before <date>  (oo,d)  f  f  f  f  f  start before <date>

start after <date>   (d,oo)  f  f  f  f  f  start after <date>
after <date>

Restriction(d,true)  (oo,d)  f  f  f  f  f  start after <date>

end before <date>    (oo,d)  f  f  f  f  f  end before <date>
before <date>

Restriction(d,false) (oo,d)  f  f  f  f  f  end before <date>

end after <date>     (d,oo)  f  f  f  f  f  end after <date>

(d1,d2) f  f  f  f  f  end first in <period>

(d1,d2) f  f  f  f  f  end last in <period>

end first in <period> (d1,d2) f  f  f  f  f  end first in <period>

end in <period>       (d1,d2) f  f  f  f  f  end last in <period>
end last in <period>
last in <period>

(d1,d2) f  f  f  f  f  start first in <period>

```

Extensions

Horizon\_Date Extension

(d1,d2) t f t start last in <period>

start first in <period> (d1,d2) t f t f start first in <period>

start in <period>

first in <period>

start last in <period> (d1,d2) t f t t start last in <period>

not in <period> (d1,d2) t t f f not in <period>

(d1,d2) t t f t not in <period>

in <period> (d1,d2) t t t f in <period>

(d1,d2) t t t t

These models have a field that is a Restriction model :  
Operation\_Plan, Load\_Plan.

specification - - a Symbol field of model Restriction

Specifies how to format Restriction values. The specification is identical to Date formats.

Default: YY-MM-DD hh:mm

Horizon\_Date - - a spec extension of model Format

Specifies how to format a Horizon\_Date value.

These models have a field that is a Horizon\_Date model :

Operation, Strategy, Problem\_Set, SIMPLE\_FIXED\_QUANTITY,  
SIMPLE\_FIXED\_TIME, WEEKLY, DUAL\_REQUEST,  
SIMPLE\_CONSOLIDATION, SHARED\_USE,  
PRODUCING\_FLOW\_CALENDAR,  
PRODUCING\_FLOW\_CALENDAR\_FILTER\_AND\_RANK, SUPPLIER,  
FIXED\_QUANTITY\_FENCED, LFL\_BOUNDED, MLFL\_BOUNDED,  
LFL\_MIN\_TIME.

time format - - a String field of model Horizon\_Date

Specifies the format of the time portion of the horizon\_date. This is the same format as described for the time portion of Date, except the granularity for horizon\_date times goes only to minutes (seconds are ignored).

Extensions

Horizon\_Date Extension

Default: hh:mm

number\_format - - a String field of model Horizon\_Date

The number format determines whether numbers are printed as numeric or ordinal values (i.e. whether 'first' should be printed as '1', '1st' or 'first'). The following character sets are supported to define the format:

'#' (digits - i.e. '1') -or- '#st' (ordinal - i.e. '1st') -or- 'st' (ordinal - full name - i.e. 'first')

Default: #st

weekday\_format - - a String field of model Horizon\_Date

The day\_of\_week format is used to print the day of the week. The supported formats are as follows: # (digit - i.e. day '1') -or- #st (ordinal - i.e. '1st day') -or- st (full name - i.e. 'first day') -or- WW - 2 character Day of week abbreviation, all caps (i.e. 'MO') Ww - 2 character Day of week abbreviation, Capitalized ww - 2 character Day of week abbreviation, lowercase WWWW - 3 character Day of week abbreviation, all caps Wwww - 3 character Day of week abbreviation, Capitalized wwww - 3 character Day of week abbreviation, lowercase WWWW\* - full Day of week name, all caps (i.e. 'MONDAY') Wwww\* - full Day of week name, Capitalized wwww\* - full Day of week name, lowercase  
Default: Wwww

day\_format - - a String field of model Horizon\_Date

Nth day of horizon format. Keywords beginning with ':' are replaced by the actual value. :DAY is required :TIME is optional

Default: :DAY day

rel\_day\_of\_month\_format - - a String field of model Horizon\_Date

[1-31]th day of Nth month of horizon format. Keywords beginning with ':' are replaced by the actual value. :DAY, :MONTH are required :TIME is optional

Default: :DAY of :MONTH month

day\_of\_month\_format - - a String field of model Horizon\_Date

[1-31]th day from today of Nth month of horizon format. Keywords beginning with ':' are replaced by the actual value. :DAY, :MONTH are required :TIME is optional  
Default: :DAY day and :MONTH month

day\_from\_end\_month\_format - - a String field of model Horizon\_Date

[1-31]th day from last day of Nth month of horizon format. Keywords beginning with ':' are replaced by the actual value. :DAY, :MONTH are required :TIME is optional  
Default: :DAY from end of :MONTH month

Extensions	List Extension
------------	----------------

**DOW\_format** - - *a String field of model Horizon\_Date*  
Nth occurrence of [1-7]th day of horizon format. Keywords beginning with ':' are replaced by the actual value. :OCCURRENCE, :DOW are required. :TIME is optional  
Default: :OCCURRENCE :DOW

**DOW\_of\_week\_format** - - *a String field of model Horizon\_Date*  
[1-7]th day of Nth week of horizon where week starts on day [1-7] format. Keywords beginning with ':' are replaced by the actual value. :DOW, :WEEK are required  
:WEEK\_START, :TIME are optional  
Default: :DOW of :WEEK week, start :WEEK\_START

**DOW\_of\_month\_format** - - *a String field of model Horizon\_Date*  
Nth occurrence of [1-7]th day of Nth month of horizon format. Keywords beginning with ':' are replaced by the actual value. :OCCURRENCE, :DOW, :MONTH are required. :TIME is optional  
Default: :OCCURRENCE :DOW of :MONTH month

**DOW\_of\_week\_month\_format** - - *a String field of model Horizon\_Date*  
[1-7]th day of [1-4]th week of Nth month of horizon where week starts on day [1-7] format. Keywords beginning with ':' are replaced by the actual value. :DOW, :WEEK are required. :WEEK\_START, :TIME are optional  
Default: :DOW, :WEEK wk, :MONTH month, start :WEEK\_START

List -- *a spec extension of model Format*

A Format which can handle all List types

**delimiter** - - *a String field of model List*  
Specifies the separator string to place between list elements. It is not put after the last element.  
Default: .

**element\_format** - - *a Symbol field of model List*  
The format to use when formatting the list elements  
Default: default

**width** - - *a Integer field of model List*  
The result when converting a list into a string will be less than this width, in characters. If the result is truncated, a '.' will be at the end of the string. For example: width = 20; list(1, 2, 3) -> "1, 2, 3" width = 10; list(1, 2, 3) -> "1, 2,..."  
Default: 100

Extensions	List Extension
------------	----------------

**truncated\_format** - - *a String field of model List*  
If the result is truncated due to the width field, this specifies the format of the stuff at the end of the result. '{0}' will contain the count of the list elements which have not been formatted. '{1}' will contain the data type of the list element. For example: if max\_end\_format = "...({0})", and width = 10, then list(1, 2, 3) -> "1, 2,...(1)"  
if max\_end\_format = "[List of {0} {1}'s]", and width = 1, then list(1, 2, 3) -> "[List of 3 Integer's]"  
Default: (+ {0})...

10.46 Field Extensions

10.46.1 field\_type extensions of model Field

SIMPLE -- a field\_type extension of model Field

A builtin field of a builtin model

SELECTOR -- a field\_type extension of model Field

A builtin field that selects a set of extension fields.

EXTENDED -- a field\_type extension of model Field

A builtin field that is only present in a subset of it's parent's models

USER -- a field\_type extension of model Field

A user defined field

get\_expression - - a Expression field of model USER

When the 'get\_expression' field is non-empty, this user-defined field will compute it's value from the expression. When the expression is evaluated, '#' will be bound to an instance of this field's model. The expression will be expected to return a the data-type indicated by this field's 'type' field.

@-Note: A field with a get\_expression, but no set\_expression is read-only.

Note: The 'default' field has no effect for fields with a get\_expression.

Note: A user-defined field normally consumes several words of storage for each model instance. That's not true for a computed field.

Default: none

set\_expression - - a Expression field of model USER

When the 'set\_expression' field is non-empty, this user-defined field will execute the given expression when the field is set. 'self' will be bound to an instance of this field's model, and 'value' will be set to the new value (the 2nd parameter to the set function). The result from the 'set\_expression' is ignored. A field with a 'set\_expression' but no 'get\_expression' will generate a Field\_Error.

Default: none

Index

Symbol Date Range	354, 718	active_product_root_forecast	450
abs_Integer	78	active_products	310
abs_Number	78	active_specific_forecasts	456
abs_Quantity	79	active_specific_products	332
abs_Time	79	active_strategies	348
accept_as_allocated	452, 457, 472	Active_Strategy	230, 348, 483, 484, 485, 486, 487, 488, 489, 490, 491, 560, 786, 787, 788, 789, 790, 791
accept_as_promised	439, 441, 445	List	348, 787, 790
accept_by	391, 413, 438	active_sub_forecasts	458, 460, 714
Acceptance	229, 352, 355, 413, 437, 438, 439, 440, 443, 717, 719, 721, 723, 724, 726, 728, 729, 731, 732, 734, 736, 737, 739, 741, 743, 744, 747, 748, 749, 751, 752, 753, 754, 755, 756, 757, 758, 759, 763, 765, 767, 768, 770, 772, 773, 775	active_top_forecasts	450
acceptance	413, 721, 723, 724, 726, 728, 729, 731, 732, 734, 736, 737, 739, 741, 743, 744, 747, 748, 749, 751, 752, 753, 754, 755, 756, 757, 758, 759, 763, 765, 767, 768, 770, 772, 773, 775	active_in	534
acceptance_inconsistent	559, 564, 755, 834, 835	actual	398
ACCEPTANCE_NOT_PLANNED	559, 563, 736, 737, 738, 825, 826	ACTUAL_OR_FORECAST	567, 871
ACCEPTANCE_PLANNED_EARLY	559, 563, 739, 740, 741, 827	actual_promises	452, 473
ACCEPTANCE_PLANNED_EXCESS	559, 563, 743, 744, 828, 829	actual_requests	451
ACCEPTANCE_PLANNED_LATE	559, 563, 738, 739, 826, 827	adjusted_target	491
ACCEPTANCE_PLANNED_SHORT	559, 563, 741, 742, 743, 827, 828	adjusted_value	490
acceptances	352, 355, 717, 719	after_fence_excise_on_hand	664
accepted	395, 438, 471	after_fence_multiple_quantity	625, 630
access	546, 548	after_fence_quantity	664
active	326, 455	after_fence_quantity_range	625, 629
active_forecasts	450	after_horizon_number	875, 876
active_generic_forecasts	456	after_horizon_quantity	876, 877
Active_Goal	230, 487, 490, 491, 561, 792, 793, 794	after_horizon_symbol	877
active_goals	487	after_horizon_time	878
active_leaf_product_forecasts	458, 460, 714	after_resolve	788, 815
active_member_forecasts	456	after_run	788, 815
active_primary_products	332	after_search	788, 815
Active_Problem	230, 486, 488, 489	after_success	548
active_problems	486	ALL	788, 815
active_product_forecasts	451	all_consume_flows	562, 804
		all_consuming_flow_plans	283
		all_consuming_operations	379
		all_flows	288
		all_items	283
		ALL_ON_TIME	304
		all_product_flows	562, 804
		all_producing_flow_plans	283
		all_producing_operations	379
		all_products	258
		all_products_and_specifies	339
		all_site_groups	340
		all_supplying_operations	309
		allocable_allocated_available	259
			457, 476

## Index

allocate_excess	677	BASIC_CALENDARS	689
allocate_quote_multiple	330	BASIC_CALENDAR	556, 588, 589, 607
allocated	470, 478	BASIC_DELAYED	556, 557, 590, 591, 607
allocated_available	475, 479	BASIC_FILTER_AND_RANK	558, 649, 650, 651, 652
allocation_policy	327, 476, 580, 581, 583, 715	BASIC_FILTER_AND_RANK	558, 649, 650, 651, 652
allocations	583	BEFORE_AND_AFTER	561, 563, 787, 788, 814, 815
Allocated_Operation	228, 297, 298, 586, 587	BEFORE_AND_AFTER	561, 563, 787, 788, 814, 815
Allocated_Product	229, 332, 335	before_horizon_number	875, 876
Allocated_Product	229, 332, 335	before_horizon_quantity	875, 877
Allocated_Products	332	before_horizon_symbol	875, 877
Allocated_Products	332	before_horizon_time	878
ALTERNATES_PRIMARY	555, 556, 586, 605, 606	before_run	787, 815
ALTERNATES_PROPORTIONAL	555, 556, 586, 587, 606	before_search	787, 815
always_override_members_committed	328	billing_address	240
always_override_members_forecasted	328	billing_city	240
and_Logical	143	billing_contact	240
announcing_goodness	485	billing_country	240
area	517	billing_fax	240
artificial	251	billing_name	239
ASAP	562, 804	billing_phone	240
ASAP_MONTHLY	562, 805	billing_postal_code	240
AT_END	555, 580	billing_state	240
at_critics	459, 713	Box	230, 247, 517, 518
ATP_Entry	230, 459, 478, 479, 713	box	247
atp_horizon	311, 451	breakpoint	163
auto_allocate	327	breakpoint	231, 554
auto_allocate_from_organization	331	breakpoint_Symbol	164
auto_run	485	bucket_fence	615
auto_run_strategy	348	bucket_list_by_date_list_void_date	615
availability_horizon	584	bucket_list_by_key_list_void_symbol	61
availability_policy	328, 584	bucket_list_list_void_date_symbol	59
available	434, 474	BUCKET_OVERSIZE	560, 565, 779, 848
available_dates	478	bucket_override	779
available_sized_time(Date_Range)	369	bucket_spec	495, 810
available_time(Date_Range)	366	bucket_start	811
available_to_promise	473	bucket_symbols_list_void	62
average_List_Integer	106	BUCKETED_ALL	562, 804
average_List_Number	107	BUCKETED_ALL_MIN_PRICE	562, 805
average_List_Percentage	107	BUCKETED_ALLOCATION	562, 805
average_List_Quantity	107	BUCKETED_ASAP	562, 806
average_List_Time	108	BUCKETED_MIN_PRICE_ASAP	562, 806
		BUCKETED_NESTED_SORT	557, 558, 621, 622, 623, 677
		bucketive_list_void_list_date_range_expression_E	58
		bucketive_expression	58
		bucket	615
		bucket(Date_Range)	495, 584
		BUFFER	565, 854
		BUFFER	228, 243, 254, 255, 256, 257, 258, 259, 260, 261, 262, 292, 356, 557, 569, 621, 623, 628, 636, 637, 638, 642, 643, 647, 648, 649, 654, 656, 657, 663, 665, 666, 667, 669, 671
		buffer	244, 252, 274, 285, 570
		buffer_levels	292, 356
		Buffer_Plan	229, 259, 349, 352, 356, 357, 358, 359, 360, 361, 362, 363, 364, 383, 558, 603, 677, 681, 685, 686,

## Index

690, 694, 697, 701, 705, 709, 716, 780, 781, 782, 783, 784	consolidation	776, 777
784	consolidation_fence	614
List	CONSOLIDATION_OVERSIZE	560, 564, 776, 777, 846
buffer_plan	CONSOLIDATION_UNDERSIZE	560, 564, 777, 846
buffer_plan (Buffer)	consolidations	619
buffer_plan (Plan)	CONSUME	556, 603, 604
buffer_plan_Filter	CONSUME_carrier	329
buffer_plans	CONSUME_FIXED	556, 598
buffer_plans (Plan)	CONSUME_flows	283
Buffer_Problem_Detector	CONSUME_PER	556, 598
buffers	consumed	472, 478
buffers_at_level (Integer)	consumed_forecast	427
	consuming_flow (Date_Range)	357
	consuming_flow_plan	385
	consuming_flow_plans	357, 364
	consuming_flows	362
	consuming_operation	667, 669, 671
	cont_Integer	164
	contains_List_Void	165
	continue	49
	continue_flow_plan_selection	635, 641, 645, 652, 661
	Control	230, 530
	convert (Measure_Unit, Measure)	512
	convert_Quantity_Quantity	80
	cos_Number	80
	cost	261, 296, 480
	count_List_Void	45
	create_consuming_operation_plan (Measure_Unit, Restriction)	359
	create_producing_operation_plan (Measure_Unit, Restriction)	359
	criteria	359
	criticism	262, 870, 871, 872, 873, 874
	cumulative_accepted	471
	cumulative_allocated	470
	cumulative_allocated_available	475, 479
	cumulative_available	474
	cumulative_committed	465
	cumulative_consumed	472
	cumulative_forecasted	462
	cumulative_planned	468
	cumulative_planned_available	475
	current	346
	current_all	789
	current_data_directory	533
	current_depth_Cell	66
	current_index_Cell	145
	CUSTOM	561, 562, 796, 799
	CUSTOMER	228, 229, 245, 355, 555, 559, 570, 719
	customer	317, 325
	customer_plan	393, 399
	CUSTOMER_RANK	566, 870
	customer_service_level	673
	customers	318, 325
	cycle_on_hand (Date)	679, 683, 688, 692, 696, 699, 703,



707, . . . . .	711	DAYS_OF_WEEK	566, 863, 864
cycle_on, hand, profile	679, 683, 688, 692, 696, 699, 703, 707, . . . . .	dec (Operation, Plan)	373
		DECREASE_CAPACITY	565, 856
		default	547
D		default_change_focus	499
daily_end	524	default_due	872, 873
daily_start	524	default_excess_on_hand	626, 630
data_directories	533	default_issued	872
Data_Directory	533	default_mean_demand	673
data_symbol	146	default_mean_demand_time_bucket	674
Date	33, 286, 287, 346, 358, 368, 376, 387, 391, 392, 395, 402, 409, 413, 414, 415, 418, 421, 422, 424, 431, 438, 439, 442, 447, 461, 464, 469, 471, 481, 484, 496, 514, 515, . . . . .	default_min_lead_time	625, 630
		default_min_time	626, 631
List	521, 529	default_number	868
date	387, 514, 515, 516	default_operation_plan_rank	347
date_accepted	392, 471	default_operation_plan_rank	622
date Activated	484	default_product_root	868
date_allocated	469	default_quantity	874
date_committed	464	default_rank	870, 871, 872, 873, 874
date_confirmed	418	default_standard_deviation_demand	674
date_forecasted	461	default_standard_deviation_lead_time	674
date_horizon, Date_Date	126	default_time	869
date_issued	392	default_value	869
date_offered	414	deficit	780, 781, 782, 784
date_queued	392	DELAY_ONLY_BASIC	556, 588, 607
Date_Range	34, 317, 346, 363, 377, 381, 383, 398, 418, 434, . . . . .	DELAY_ONLY_FIXED	535, 536, 588, 606
	436, 441, 461, 478, 480, 523, 528, 567, 883	delta_Void	151
List	451, 495, 584	DELIVER	536, 604, 605
date_range, Date_Date	127	delivery	229, 417, 438, 440, 441, 442, 443, . . . . .
date_range, Date_Time	127	Delivery_Acceptance	444, 448, 563, 721, 723, 724, 726, 727, 729, 731, 732, . . . . .
date_range_String	128		734, 735, 737, 739, 741, 742, 744, 747, 748, 749, 751, 752, 753, 757, 758, 759, 763, 765, 766, 768, 770, 771, 772, 753, 757, 758, 759, 763, 765, 766, 768, 770, 771, 773, . . . . .
date_range_String_Symbol	128	delivery_acceptance	417, 721, 723, 724, 726, 727, 729, . . . . .
date_String	116		731, 732, 734, 735, 737, 739, 741, 742, 744, 747, 748, 749, 751, 752, 753, 757, 758, 759, 763, 765, 766, 768, 770, 773, . . . . .
date_String_Symbol	117	DELIVERY_AVAILABLE_To_Promise	229, 417, 424, 425
DATES	562, 810, 811	List	229, 394, 400
dates	363, 377, 381, 383, 434, 436, 480	delivery_city	229, 394, 400
dates (Date_Range)	521, 529	delivery_contact	229, 394, 400
day	864, 865, 866, 867	delivery_country	229, 394, 400
day_buckets	810	delivery_date	424
day_format	894	delivery_dates	461
day_format	894	delivery_discount	419
day_from_end_month_format	894	delivery_fax	229, 394, 400
day_lat, power	859	delivery_name	229, 394, 400
DAY_OF_LAST_WEEK_OF_MONTH	566, 866	delivery_naming	391, 803
DAY_OF_MONTH	566, 864	delivery_not_confirmed_problem	443
day_of_month_Date	122	delivery_phone	229, 394, 400
day_of_month_format	894		
day_of_week	577		
day_of_week_Date	122		
DAY_OF_WEEK_OF_MONTH	566, 865		
DAY_OF_YEAR	566, 866		
day_of_year_Date	122		
day_pattern	525, 863, 864, 865, 866, 867		
DAYS	562, 810		
days	864		
days_Date_Range	135		
days_Date_Range_Time	136		

delivery_plan	408, 429, 445, 721, 723, 725, 726, 728, 729,	DOW_format	.....	89.
731, 733, 734, 736, 738, 739, 741, 743, 744, 763, 765,		DOW_of_month_format	.....	899.
767, .....	391, 413, 803, 808	DOW_of_week_format	.....	899.
delivery_policy	.....	DOW_of_week_month_format	.....	899.
delivery_postal_code	.....	DOWSTREAM	.....	565, 855
delivery_price	.....	downstream_bucket_pad	.....	616
Delivery_Promise	229, 398, 413, 416, 417, 418, 419, 420,	downstream_flow_plans	.....	38.
421, 422, 423, 424, 425, 426, 432, 441, 562, 721, 722,		downstream_joi_flows	.....	388.
724, 726, 727, 729, 730, 732, 734, 735, 737, 739, 740,		downstream_pad	.....	611
742, 744, 747, 748, 749, 750, 752, 753, 756, 757, 759,		drawing_id	.....	250
763, .....	764, 766, 768, 770, 771, 773, 774, 809	DUAL_REQUEST	.....	555, 578, 579, 580
delivery_promise	398, 441, 721, 722, 724, 726, 727, 729,	due	.....	398, 418, 441, 444
730, 732, 734, 735, 737, 739, 740, 742, 744, 747, 748,		DUE_DATE	.....	567, 873
749, 750, 752, 753, 756, 757, 759, 763, 764, 766, 768,				
770, .....	771, 773, 774	E		
delivery_promise_filler	.....	EARLIEST	.....	562, 800
DELIVERY_PROMISE_NOT_COORDINATED	.....	EARLIEST_PROBLEM_START	.....	561, 563, 791, 818
564, .....	757, 758, 838	echo_Logical	.....	150
DELIVERY_PROMISE_OVERPRICED	.....	echo_String	.....	150
delivery_promises	.....	citable	.....	536, 543, 548
Delivery_Request	229, 391, 397, 398, 399, 400, 401, 402,	effective	.....	588
403, 404, 405, 411, 417, 562, 721, 722, 724, 726, 727,		effective_alienmats	.....	588
729, 730, 732, 734, 735, 737, 739, 740, 742, 744, 747,		EFFECTIVE_CALENDAR	.....	555, 556, 587, 588, 606
748, 749, 750, 752, 753, 756, 757, 758, 763, 764, 766,		Effective_Calendar_Operation	.....	228, 298, 299, 588
768, .....	769, 771, 773, 774, 804, 805, 806, 807	effective_dates	.....	317, 317
delivery_request	417, 721, 722, 724, 726, 727, 729, 730,	efficiency	.....	265, 270, 275, 366, 610, 617, 798
732, 734, 735, 737, 739, 740, 742, 744, 747, 749,		efficiency_average(Date_Range)	.....	365
750, 752, 753, 756, 757, 758, 763, 764, 766, 768, 769,		efficiency_average_at_skill(Skill), Date_Range)	.....	366
771, .....	773, 774	efficiency_calendar	.....	610, 617, 798
DELIVERY_REQUEST_NOT_COORDINATED	.....	efficiency_level(Date)	.....	270, 275
564, .....	756, 757, 838	efficiency_profile(Date_Range)	.....	275, 365
delivery_requests	.....	efficiency_profile_at_skill(Skill), Date_Range)	.....	366
delivery_state	.....	element, formal	.....	895
demand_safety_stock	.....	element_List_Void_Integer	.....	50
description	233, 237, 246, 250, 255, 264, 274, 281, 303,	empty	.....	518
308, 313, 317, 324, 338, 346, 350, 391, 480, 493, 494,		enable	.....	164
498, .....	520, 523, 536, 542, 545, 547, 550, 551	enable_Symbol	.....	164
detector	.....	enable_Date_Date_Range	.....	128
deterministic_problem_selection	.....	enclose_Date_Date_Range	.....	129
deterministic_resolvers	.....	enclose_Quantity_Quantity_Range	.....	102
dimensions	.....	enclose_Quantity_Range_Quantity_Range	.....	102
directory	.....	enclosing_day_Date	.....	125
disable	.....	end_Date_Range	.....	129
disable_Symbol	.....	end_item	.....	627, 632
discount	.....	end_location	.....	290
district	.....	end_setup	.....	290
do_.....	511	ending_on_hand	.....	363
do_after	.....	engine_Expression	.....	159
do_before	.....	engine_name_	.....	148
do_before_resolve	.....	entries	.....	456, 520
do_file_echo_String	.....	entries_consumed	.....	457
do_file_fast_String	.....	entries_members_consumed	.....	457
do_no_move_out_forecast	.....	ENTRY_DATE	.....	521, 529
Domain	.....	entry(Date)	.....	567, 874
domains	.....	entry_value	.....	520, 528, 868, 869, 875, 876, 877, 878
done	.....	error_input	.....	551
	.....	error_value	.....	551

## Index

evaluable_Expression	160	filter_List_Void_Expression_Integer	45
evaluation	788, 789, 816, 817	find_List_Void_Void	50
EVEN	561, 797	find_or_create_List_Void_Void	51
EVERY_N_DAYS	566, 863	find_or_nomenclature_List_Void_Void	50
EVERYDAY	566, 863	finite_Date	1222
exact_String_String	75	finite_Number	79
exact_Symbol_Symbol	77	first_day_of_week	311, 810
EXCESS	785	first_List_Void	52
EXCESS_ON_HAND	560, 565, 784, 785, 833	first_value_Typed_Value	151
EXCESS_ON_HAND	649, 650, 656, 658, 663, 665	FIXED	555, 556, 557, 561, 562, 585, 597, 610, 611, 617, 708, 803, 808
EXCESS_ON_HAND (Date)	680, 684, 689, 693, 696, 700, 704, 708, 711	FIXED_COUNT	557, 612
EXCESS_ON_HAND_AT_END	560, 565, 785, 854	fixed_efficiency	610, 617, 798
EXCESS_ON_HAND_CALENDAR	626, 631	FIXED_QUANTITY	557, 558, 612, 654, 655, 694, 695, 696, 697
EXCESS_ON_HAND_PROFILE	680, 684, 689, 692, 696, 700, 704, 707, 711	fixed_quantity	597, 598, 599
EXCESS_QUANTITY	783	FIXED_QUANTITY_FENCED	558, 663, 664, 705, 706, 707
EXCESS_LINE	784	FIXED_SPLIT	555, 583
EXCLUSIVE_USE	557, 614, 619	FIXED_TIME	556, 558, 589, 607, 665, 666, 709, 710, 711, 712
EXCISE	487, 499, 786, 787, 788, 789, 790, 813, 814, 815, 816, 817	Flow	228, 282, 292, 293, 383, 556, 598, 599, 600
EXCISE_Void	146	List	283
EXISTS_WITHOUT_ERRORS_Void	147	Flow	228, 262, 566, 623, 870, 871, 872, 873, 874
EXP_Number	147	Flow_Criterion	283
EXP_Number	377		383
expiration_policy	327, 580		
EXPLICIT_Slice	228, 314, 315	FLOW_LIMIT_CALENDAR_FILTER_AND_RANK	
explicit_slices	314		
EXPORT_starting_void	147		
Expression	34, 287, 347, 501, 526, 546, 548, 632, 633, 634, 635, 636, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 666	Flow_Plan	229, 379, 383, 384, 385, 386
Expression	646, 667, 650, 651, 652, 654, 659, 660, 661, 662, 666, 667, 653, 641, 646, 653, 662, 660	List	356, 357, 359, 360, 364, 379, 384
Expression	612, 813, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 857, 897	flow_plan_filter	634, 640, 644, 652, 660
EXPRESSION	857	flow_plan_move_restriction	635, 641, 646, 653, 662, 660
EXTENDED	567, 897	flow_plan_rank	634, 640, 645, 652, 660
EXENSIBLE	546	flow_plan_resize_quantity_range	635, 641, 646, 653, 661
Extension_Selector	231, 546, 548, 550	flow_plan_selection_end	633, 639, 644, 651, 659
Extension_selector	548	flow_plan_selection_interval	633, 639, 644, 651, 659
extension_selectors	546	flow_plan_selection_start	632, 638, 643, 650, 659
EXTRACTIONS	548, 550	flow_plan_split_restriction	630, 642, 647, 654, 662
		flow_plans	356, 379
family	250	focus	488, 504, 506
FCFS	555, 580	focus_beyond_target	508
FEASIBILITY	561, 565, 792, 838	focus_to_target	508
feasible	261, 296, 480	focus_value	491
feasible_focus	504	for_each_List_Void_Expression	46
feasible	502, 625, 629, 648, 664	Forecast	229, 349, 393, 427, 436, 450, 451, 453, 454, 455, 456, 457, 459, 461, 476, 478, 479, 558, 713
Field	231, 546, 547, 548, 549, 551, 567, 897	List	427, 450, 451, 453, 456, 458, 459, 460, 713, 714
List	550	forecast	393, 436
Field_Error	231, 551, 552	forecast (Product)	349
Field_Type	548, 897	forecast (Symbol)	450
Fields	493, 546	Forecast_Entry	230, 398, 436, 456, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 65

## Index

465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, . . . . .	538, 715, 760, 761	398, 436, 760, 761	311, 451	318, 326, 476, 574, 575, 577, 578, 580	476	476	462	450	885, 886, . . . . .	889, 890, 891, 892, 893, 895	534	362	49	399, 418, 441, 807, 809, 812	492, 532	562, 563, 807, 809, . . . . .	812	34	231, 553	155	456	334	331	148	149	897	149	861	499	714	332	543	517	381	382	584	811	579, 614, 615, 625, 629, 648, 664, 668, 670, 893, 894, 895	120	123	77	35	145	160	147	482		
forecast_currency	forecast_horizon	forecast_policy	forecast_promises	forecast_requests	forecasted	forecastis	Format	886, . . . . .	formals	formatted	found_List_Void_Void	fullname_policy	full_name	FULL_QUANTITIES_OF_ALL_ITEMS	Func	Function	functions_Symbol	Generic_forecasts	Generic_Product	generic_products	get_color_String	get_drag_string_Integer	get_expression	getenv_String	goal	goals	GROUP	Groups	H	handles_Type	height	hint	hint_on	HORIZON	horizon	horizon_230_311_494_495_496_562_584_615_622_810	horizon_346_504_528_622	Horizon_Bucket_Start	Horizon_Date_35_286_502_504_567_575_576_578_579_614_615_625_629_648_664_668_670_893_894_895	hour_date_String	hour_date	iclar_Integer	ID	id_Void	identifier	if_Logical_Void_Void	import_string_void	in_category_Problem_Category
in_New Date_Range	in_New plans	IN_FRONT	inc Operation_Plan	inc_new_data_layout String_String	include	INCONSISTENT_OPLAN	INCREASE_CAPACITY	index_String_String	index_String_String_Integer	INDIVIDUAL	INFINITE	INFINITE_USE	inspect_void	inspect_int	inspect_int_int	inspect_int	inspect_int	Integer_Number	Integer_Percentage	integer_String	integer_String_Symbol	integers_Integer_Integer	interaction	interchangeable Configuration	interruptible	intersect Date_Range_Date_Range	intersect Date_Range_Date_Range	interval	interval_Quantity_Range	intra_horizon_number	intra_horizon_quantity	intra_horizon_symbol	intra_horizon_time	is_directory_String	is_file_String	is_variable_String	issued	item_228_243_244_249_250_251_252_253_255_300_306_321_406_428_445_561_569_570_592_593_605_796_800_801	item_demand_List	item_243_251_304_324_569_605_800	item_225_300_306_321_406_428_445_592_593_605_800	item_Acceptance	721_722_724_725_727_729_730_733_735_737_738_740_742_744_749_750_752_759_762_764_766_768_769_771_772_774	item_acceptance	426_721_722_724_725_727_729_730_732_733_735_737_738_740_742_744_749_750_752_762_764_766_768_769_771_772_774			

item\_acceptance\_filter ..... 826, 827, 828  
 item\_acceptances ..... 441  
 item\_aip ..... 425, 427  
 item\_available\_to\_promise ..... 229, 427, 433, 434, 435, 437  
 List ..... 425  
 item\_group ..... 228, 243, 302, 303, 304, 305, 306, 569  
 List ..... 243, 569  
 item\_group ..... 305  
 item\_groups ..... 243, 569  
 item\_name ..... 407, 418, 426, 427, 428, 429, 430, 431, 432, 433, 434, 444, 604, 720, 722, 724, 725, 727, 729, 730, 732, 733, 735, 737, 738, 740, 742, 743, 749, 750, 751, 762, 764, 766, 768, 769, 771, 772, 774  
 List ..... 473, 476  
 item\_promise ..... 407, 444, 720, 722, 724, 725, 727, 729, 730, 732, 733, 735, 737, 738, 740, 742, 743, 749, 750, 751, 762, 764, 766, 768, 769, 771, 772, 774  
 item\_promise\_filter ..... 823, 824, 825, 833, 837, 843  
 ITEM\_PROMISE\_OVERPRICED ..... 559, 751, 752  
 item\_request ..... 229, 398, 405, 406, 407, 408, 409, 410, 411, 426, 604, 608, 720, 722, 724, 725, 727, 728, 730, 732, 733, 735, 737, 738, 740, 742, 743, 749, 750, 751, 762, 764, 766, 767, 769, 771, 772, 774  
 List ..... 354, 476, 718  
 item\_request ..... 426, 608, 720, 722, 724, 725, 727, 728, 730, 732, 733, 735, 737, 738, 740, 742, 743, 749, 750, 751, 762, 764, 766, 767, 769, 771, 772, 774  
 item\_request\_filter ..... 820, 821, 822, 836, 839, 840, 841, 844  
 item\_requests ..... 398  
 items ..... 243, 244, 320, 324, 569, 570  
  
 J  
 justify\_string\_int\_string ..... 71  
  
 K  
 key\_names\_list\_void ..... 56  
 key\_symbol\_symbol\_logical ..... 54  
 key\_values\_list\_void\_symbol ..... 56  
 keyed\_element\_symbol ..... 63  
 keyed\_list\_list\_void\_expression ..... 62  
 keys\_list\_void ..... 63  
  
 L  
 language ..... 395, 402, 409, 414, 421, 431, 439, 442, 447, 481  
 last\_change\_after\_activated ..... 502  
 last\_list\_void ..... 52  
 Layout ..... 230, 534, 539  
 layout ..... 534  
 local\_categories ..... 493  
 local\_string ..... 68  
 local\_string\_integer ..... 68  
 local\_string\_string ..... 69  
 len\_string ..... 74  
  
 level ..... 257, 454, 459, 713  
 LFL\_BOUNDED ..... 558, 667, 668, 669  
 LFL\_SIMPLE ..... 558, 666, 667  
 limit\_date\_date\_range ..... 131  
 limit\_date\_range\_date\_range ..... 131  
 limit\_quantity\_quantity\_range ..... 100  
 limit\_quantity\_range\_quantity\_range ..... 100  
 LINEAR ..... 556, 597  
 linear\_quantity ..... 597  
 LINK ..... 228, 229, 242, 243, 244, 352, 353, 354, 355, 555, 559, 568, 569, 570, 716, 717, 718, 719  
 List ..... 567, 893, 896  
 list ..... 52  
 list\_index\_list\_void\_void ..... 51  
 list\_price ..... 418, 428  
 In\_Number ..... 82  
 Load ..... 228, 282, 288, 289, 290, 291, 381, 556, 596  
 load ..... 381  
 load\_data\_layouts (String) ..... 533  
 load\_files ..... 534  
 List ..... 367  
 Load\_Plan ..... 229, 379, 381, 382  
 load\_plans ..... 367, 379  
 load\_policy ..... 268, 367, 614, 619  
 Load\_Size ..... 228, 290, 291, 556, 597  
 load\_size\_usage ..... 291, 597  
 load\_size\_and\_time (Date, Range) ..... 370  
 load\_size\_and\_time (Date, Range) ..... 369  
 load\_size ..... 290  
 load\_size (Date, Range) ..... 367  
 load\_time (Date, Range) ..... 367  
 loading\_buffers ..... 274  
 loading\_operations ..... 274  
 loads ..... 282  
 List ..... 243, 247, 569  
 Location ..... 228, 242, 246, 247, 248, 249, 256, 264, 290, 568  
 location ..... 256, 264  
 locations ..... 242, 568  
 lock\_allocated ..... 470  
 lock\_end\_on\_hand ..... 364  
 lock\_on ..... 382  
 lock\_retain\_from\_allocated ..... 468  
 locked\_as\_planned ..... 378  
 locks ..... 499  
 log\_number ..... 82  
 log\_number\_number ..... 82  
 log10\_number ..... 83  
 Logical ..... 35, 238, 242, 247, 251, 261, 282, 292, 293, 296, 301, 303, 309, 326, 327, 328, 330, 331, 332, 338, 339, 362, 364, 375, 378, 382, 383, 398, 402, 407, 408, 421, 430, 442, 446, 455, 460, 464, 467, 468, 470, 476, 480, 482, 484, 485, 486, 488, 489, 498, 502, 504, 511, 518, 533, 534, 536, 543, 546, 548, 567, 568, 582, 586, 587, 604, 623, 627, 628, 632, 714, 786, 870, 871, 872, 873, 874, 879, 880  
 logical\_string ..... 76  
 logical\_string\_symbol ..... 76

Lot ..... 229, 358, 361, 362, 385  
 lot ..... 385  
 Lot ..... 358, 362, 385  
 Lot\_Flow ..... 229, 362, 384, 385, 386  
 LOT\_NOT\_CONSUMED ..... 560, 565, 783, 851, 852  
 LOT\_NOT\_PRODUCED ..... 560, 565, 783, 851, 852  
 LOT\_OVER\_CONSUMED ..... 560, 565, 782, 851  
 LOT\_OVER\_PRODUCED ..... 560, 565, 783, 784, 852  
 lots ..... 358, 384  
 lots\_unchecked ..... 251  
 LOW\_ON\_HAND ..... 560, 565, 784, 852, 853  
 low\_on\_hand ..... 780, 781, 782, 784  
 low\_on\_hand (Date) ..... 681, 684, 690, 693, 697, 701, 705, 708, 712  
 low\_on\_hand\_profile ..... 680, 684, 689, 693, 697, 701, 704, 708, 712  
 lower\_string ..... 69  
  
 M  
 maintenance ..... 267, 370, 611  
 make\_type\_void\_type ..... 151  
 managed ..... 242, 568  
 MANUAL ..... 558, 561, 563, 671, 786, 813  
 margin\_large (Date, Date) ..... 238  
 matching\_count\_string\_string ..... 75  
 matching\_forecast ..... 427  
 max\_all ..... 789, 790, 816, 817, 855  
 max\_bucket\_load ..... 616  
 MAX EFFICIENCY ..... 561, 797  
 max\_heat ..... 500  
 max\_list\_date ..... 116  
 max\_list\_integer ..... 108  
 max\_list\_number ..... 108  
 max\_list\_percent ..... 109  
 max\_list\_quantity ..... 109  
 max\_list\_time ..... 109  
 max\_operation\_count ..... 612  
 max\_price ..... 399, 407  
 max\_quantity\_range ..... 101  
 max\_rank ..... 857  
 max\_resolve\_count ..... 500  
 max\_run\_time ..... 500  
 max\_size ..... 272, 612  
 max\_stable\_resolve\_count ..... 500  
 max\_stable\_time ..... 500  
 max\_target (Date) ..... 678, 681, 687, 690, 694, 698, 702, 705, 709  
 max\_target\_profile ..... 677, 681, 686, 690, 694, 698, 701, 705, 709  
 MAXIMIZE\_PROFIT ..... 561, 566, 794, 860, 861  
 MAXIMIZE\_REVENUE ..... 561, 566, 794, 861  
 mean\_demand\_calendar ..... 674  
 mean\_lead\_time\_calendar ..... 674  
 Measure ..... 35, 511  
 measure\_quantity ..... 83  
 measure\_string ..... 83  
 measure\_string\_symbol ..... 84  
  
 Measure\_Unit ..... 36  
 member ..... 356  
 member\_forecasts ..... 455  
 member\_of (Explicit, Site) ..... 238  
 member\_of (Seller) ..... 309  
 member\_of (User) ..... 533  
 MEMBER\_RANK ..... 555, 558, 581, 582, 583, 715  
 members ..... 238, 309, 350, 449, 533  
 members\_accepted ..... 472  
 members\_allocated ..... 471  
 members\_available ..... 474  
 members\_committed ..... 466  
 members\_consumed ..... 473  
 members\_forecasted ..... 463  
 members\_planned ..... 469  
 memory\_size ..... 154  
 mid\_string\_integer ..... 69  
 mid\_string\_integer\_integer ..... 70  
 mid\_x ..... 517  
 mid\_y ..... 518  
 min\_all ..... 789, 790, 816, 817, 855  
 min\_bucket\_overflow ..... 848  
 min\_cost ..... 504  
 min\_deficit ..... 849, 850, 853  
 min\_delivery\_lead\_time ..... 317, 324  
 min\_effort ..... 821, 824, 827, 832, 839, 842  
 min\_excess ..... 822, 825, 828, 833, 841, 843, 853, 854  
 min\_excess\_time ..... 852  
 min\_heat ..... 500  
 min\_lateness ..... 820, 823, 826, 831, 839, 841  
 min\_list\_date ..... 116  
 min\_list\_integer ..... 109  
 min\_list\_number ..... 110  
 min\_list\_percent ..... 110  
 min\_list\_quantity ..... 110  
 min\_list\_time ..... 111  
 min\_load ..... 615  
 min\_load\_fence ..... 615  
 min\_on\_hand ..... 649, 650, 655, 657, 658, 663, 665  
 min\_on\_hand\_calendar ..... 626, 650  
 min\_overflow\_and\_time ..... 847  
 min\_overflow\_time ..... 847  
 min\_overflow ..... 847  
 min\_oversize ..... 847  
 min\_quantity ..... 317, 324  
 min\_quantity\_range ..... 101  
 min\_rank ..... 857  
 min\_shortness ..... 821, 824, 828, 832, 840, 842  
 min\_size ..... 272  
 min\_time ..... 502, 649, 650, 655, 657, 658, 663, 665  
 min\_time\_calendar ..... 627, 631  
 min\_underload ..... 848  
 MINIMIZE\_COST ..... 561, 566, 794, 860  
 MINIMIZE\_LATENCY ..... 561, 566, 793, 859  
 MINIMIZE\_PROBLEM\_COUNT ..... 561, 566, 792, 858, 859  
 MINIMIZE\_PROBLEMS ..... 561, 566, 792, 858, 859  
 MINIMIZE\_SHORTNESS ..... 561, 566, 793, 794, 860

## Index

minute, Date ..... 123  
 MFL, BOUNDED ..... 558, 669, 670, 671  
 model, file ..... 149  
 model, type ..... 231, 545, 546, 547, 549, 550  
 model\_type\_void ..... 153  
 Money ..... 36, 238, 261, 296, 399, 401, 407, 408, 418, 419, 420, 428, 429, 436, 480, 504, 585  
 money, String ..... 84  
 money, String-Symbol ..... 85  
 month ..... 123  
 month, Date ..... 867  
 MONTHS ..... 562, 810  
 months ..... 864, 865, 866  
 months, Date, Range ..... 376, 603, 604, 605  
 motive ..... 137  
 motive, promise ..... 604  
 motive, request ..... 604  
 move (Load, Plan, Logical, Logical, Logical) ..... 368  
 MOVE\_IN ..... 565, 855  
 MOVE\_OUT ..... 565, 855  
 move\_to, alternate (Operation, Plan, Operation, Quantity) ..... 606  
 MULTI\_DIMENSION ..... 557, 613, 614  
 multi\_key\_bucketize\_element\_List\_Void\_Integer\_Date\_Symbol ..... 57  
 multi\_key\_bucketize\_List\_Void\_List\_Void\_List\_Date\_Range\_Expression\_Expression ..... 56  
 multi\_key\_Symbol\_Void ..... 54  
 multi\_keyed\_element\_List\_Void\_Integer\_Symbol\_Sym\_bol ..... 55  
 multi\_keyed\_list\_List\_Void\_List\_Void\_Expression ..... 55  
 MULTIPLE ..... 558, 656, 657, 697, 698, 699, 700, 701  
 MULTIPLE\_FILTER\_AND\_RANK ..... 558, 657, 658, 659, 660, 662, 701, 702, 703, 704, 705  
 multiple\_quantity ..... 624, 629, 656, 658  
 must\_resolve ..... 489, 504  
  
**N**  
 name ..... 233, 237, 246, 250, 255, 264, 272, 274, 281, 289, 291, 292, 303, 307, 313, 317, 323, 338, 345, 361, 373, 390, 398, 406, 455, 492, 494, 498, 507, 520, 522, 532, 542, ..... 545, 547, 550  
 name\_from\_customer ..... 394, 399  
 name\_type ..... 542  
 near\_time ..... 599, 600  
 NEEDS\_RELEASE ..... 559, 564, 745, 830  
 NEGATIVE\_ATP ..... 560, 759, 760  
 NEGATIVE\_ON\_HAND ..... 560, 565, 780, 849  
 NEGATIVE\_ON\_HAND\_AT\_END ..... 560, 565, 781, 782, 850  
 NEGATIVE\_PLANNED\_ATP ..... 560, 760  
 next ..... 164  
 next\_gap (Load, Plan) ..... 368  
 next\_integer ..... 164  
 next\_release\_number ..... 287  
 NO\_PROBLEMS ..... 560, 563, 786, 813  
 NONE ..... 555, 572, 585  
  
 nonexistent ..... 152  
 not\_Logical ..... 143  
 not\_planned ..... 402, 408, 421, 430, 442, 446  
 now ..... 117  
 nih ..... 863, 865  
 NUMBER ..... 566, 567, 862, 868, 875  
 Number ..... 37, 237, 262, 299, 308, 311, 324, 335, 375, 376, 399, 418, 482, 485, 488, 490, 491, 500, 509, 514, 517, 518, 525, 536, 567, 590, 787, 792, 793, 794, 810, 857, 858, 859, 860, 861, 862, 868, 870, 871, 872, 873, 874, 875, ..... 876, 877, 883, 884  
 number ..... 862  
 number, format ..... 894  
 number, Integer ..... 85  
 number, of, weeks, Date, Date ..... 124  
 number, Percentage ..... 85  
 NUMBER\_QUANTITY ..... 566, 567, 862, 868, 876, 877  
 number, Quantity ..... 85  
 number, String ..... 86  
 number, String-Symbol ..... 86  
 NUMBERED ..... 562, 803  
  
**O**  
 offer, promise ..... 425, 434, 436  
 offered ..... 415  
 offered\_quantity ..... 436  
 on\_hand ..... 357  
 ON\_HAND\_CALENDAR ..... 557, 558, 637, 638, 685  
 ON\_HAND\_CALENDAR\_FILTER\_AND\_RANK ..... 557, 558, ..... 638, 639, 640, 641, 642, 685  
 on\_hand\_date ..... 358  
 on\_hand\_date (Date) ..... 358  
 on\_hand\_profile ..... 357  
 ON\_TIME ..... 562, 563, 804, 807, 809, 812  
 ONE ..... 556, 562, 596, 810  
 OPERATION ..... 564, 830, 831  
 List ..... 244, 258, 259, 274, 283, 284, 570  
 Operation ..... 228, 243, 252, 276, 281, 282, 283, 284, 285, 286, 287, 288, 290, 292, 293, 295, 296, 297, 298, 299, 300, 375, 555, 569, 586, 587, 588, 589, 590, 591, 593, 594, 621, 628, 632, 649, 650, 655, 657, 658, 659, 664, 666, ..... 667, 669, 671, 800  
 operation ..... 297, 298, 299, 375, 800  
 operation\_levels ..... 244, 570  
 List ..... 285, 349, 373, 379, 387, 619, 775, 776, 777  
 Operation\_Plan ..... 229, 352, 359, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 387, 408, 429, 430, 445, 556, 603, 604, 605, 606, 607, 608, 716, 721, 723, 725, 726, 728, 729, 731, 733, 734, 736, 738, 739, 741, 743, 744, 745, 746, 762, 763, 764, 765, 766, 767, 768, 769, ..... 771, 772, 774  
 operation\_plan ..... 387, 745, 746  
 operation\_plan\_filter ..... 829, 830, 831, 845  
 OPERATION\_PLAN\_RANK\_EXPRESSION ..... 565, 857  
 OPERATION\_PLAN\_RANK\_RANGE ..... 565, 857  
 operation\_plans ..... 352, 373, 387, 716, 775, 776, 777  
 operation\_plans (Operation) ..... 349

## Index

operation\_plans (Plan) ..... 285  
 Operation\_Problem\_Decider ..... 228, 285, 295, 296, 556  
 Operation\_State ..... 229, 352, 386, 387, 388, 562, 716, 775, 800, ..... 801  
 operation\_state ..... 775  
 operation\_status ..... 352, 716  
 operations ..... 243, 569  
 operations\_at\_level (Integer) ..... 244, 570  
 option\_String ..... 149  
 or\_Logical\_Logical ..... 144  
 order\_lead\_time ..... 592, 593  
 order\_quantity ..... 574  
 ORDERED\_RESOLVES ..... 561, 563, 790, 817, 818  
 Ordered\_Sub\_Strategy ..... 230, 509, 814, 818  
 organization ..... 237, 308, 449, 533  
 organization\_plan ..... 350  
 out\_flow (Date, Range) ..... 360  
 out\_flow\_plans ..... 360  
 OVER\_COMMITTED ..... 560, 760, 761  
 OVER\_CONSUMED ..... 560, 761  
 OVER\_FLOW\_LIMIT ..... 560, 565, 781, 849, 850  
 OVERLOAD ..... 560, 565, 778, 846, 847  
 overload\_sid\_time ..... 778  
 overload\_time ..... 778  
 override\_members\_committed ..... 467  
 override\_members\_forecasted ..... 464  
 OVERSIZE ..... 560, 565, 778, 779, 847  
 oversize ..... 779  
 oversize (Symbol) ..... 777  
 oversize\_dimensions ..... 777  
 owner ..... 241, 249, 253, 259, 261, 262, 269, 270, 272, 274, 276, 287, 290, 291, 293, 296, 298, 299, 300, 301, 304, 305, 306, 312, 314, 315, 319, 320, 321, 333, 334, 335, 336, 340, 342, 344, 349, 351, 360, 362, 364, 372, 373, 380, 382, 384, 386, 388, 396, 404, 411, 415, 423, 425, 432, 434, 437, 439, 443, 448, 452, 457, 476, 479, 482, 487, 489, 491, 496, 504, 506, 507, 508, 509, 510, 513, 525, ..... 527, 536, 543, 549, 550  
 owner\_model ..... 545  
 owner\_Typeed\_Value ..... 151  
  
**P**  
 Pathname ..... 37, 533, 536, 542  
 PER\_ALLOCATED ..... 555, 581  
 PER\_COMMITTED ..... 555, 581  
 per\_next\_cending\_on\_hand ..... 622  
 Percentage ..... 37, 270, 275, 297, 298, 336, 341, 343, 365, 366, 377, 419, 428, 488, 499, 504, 506, 508, 510, 515, 567, 581, 582, 583, 599, 600, 610, 615, 616, 617, 622, 673, ..... 777, 779, 780, 798, 847, 848, 885, 886  
 percentage ..... 297, 298  
 percentage\_Integer ..... 86  
 percentage\_Number ..... 87  
 percentage\_Quantity ..... 87  
 percentage\_String ..... 87  
 percentage\_String\_Symbol ..... 88  
 permanent ..... 485  
  
 phantom ..... 292  
 Plan ..... 229, 234, 345, 346, 347, 348, 349, 350, 351, 449, 452, 480, 482, 483, 487, ..... 439, 441, 446  
 plan\_as\_accepted ..... 414, 420, 430  
 plan\_as\_promised ..... 393, 402, 408  
 plan\_requested ..... 403, 422, 443  
 plan\_problems ..... 393, 401, 408, 414, 420, 429, 438, 441, 445  
 plan\_to\_satisfy ..... 353, 717  
 plan\_to\_satisfy\_all\_promises ..... 353, 717  
 plan\_to\_satisfy\_all\_requests ..... 353, 717  
 plan\_to\_satisfy\_queued\_requests ..... 353, 717  
 plan\_to\_satisfy\_unanswered\_requests ..... 353, 717  
 planned ..... 468  
 planned\_available ..... 475  
 PLANNED\_BEFORE\_CURRENT ..... 559, 563, 744, 745, 829  
 planned\_date\_problem ..... 410, 432, 447  
 planned\_early ..... 402, 409, 421, 430, 442, 446  
 planned\_exceptions ..... 409, 430, 446  
 planned\_for\_promise ..... 604  
 planned\_late ..... 402, 409, 421, 430, 442, 446  
 planned\_quantity\_problem ..... 411, 432, 447  
 planned\_start ..... 409, 430, 446  
 planning\_buffers ..... 285  
 planning\_resources ..... 234  
 plans ..... 582  
 pool\_allocation ..... 715  
 pooled\_accepted ..... 715  
 pooled\_allocated ..... 715  
 pooled\_allocated\_available ..... 715  
 pooled\_available ..... 715  
 post\_load\_delay ..... 591  
 power\_Number\_Number ..... 88  
 pre\_load\_delay ..... 591  
 Predicate ..... 37  
 PREFER\_PRIMARY ..... 561, 797  
 preferred\_measure ..... 511  
 previous\_gap (Load, Plan) ..... 368  
 price ..... 428, 436, 585  
 price\_policy ..... 328, 585  
 PRIMARY ..... 561, 797  
 primary ..... 797  
 primary\_products ..... 332  
 print\_report\_variables ..... 166  
 print\_variables ..... 165  
 Priority ..... 37  
 List ..... 347, 354, 359, 370, 380, 395, 403, 410, 415, 422, 431, 439, ..... 443, 447, 452, 486, 718  
 Problem ..... 230, 347, 403, 410, 411, 422, 431, 432, 443, 447, 480, 481, 482, 488, 559, 720, 721, 723, 725, 726, 728, 730, 731, 733, 734, 736, 738, 739, 741, 743, 744, 745, 746, 747, 748, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 765, 767, 769, 770, 772, 773, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785  
 problem ..... 488

## Index

problem_categories	348, 355, 359, 371, 380, 486, 719	product_groups	310
List	348, 355, 359, 371, 380, 486, 493, 719	Product_Item	228, 320, 332, 311
Problem_Category	230, 492, 493	product_List_Integer	111
problem_count	792	product_List_Number	1111
problem_detectors	257, 268, 285	product_List_Percentage	112
problem_filter	813, 820, 821, 822, 823, 824, 825, 826, 827,	product_List_Quantity	11212
828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838,		Product_Root	228, 310, 316, 317, 318, 319, 320, 323, 555, 574, 575, 577, 578, 622
839, 840, 841, 842, 843, 844, 845, 847, 849, 850, 851,		product_root	323
852,	853, 854	product_root_forecast	450
problem_selection	487, 500, 790, 791, 818	product_roots	310
Problem_Sel	230, 498, 502, 503, 504, 563, 820, 821, 822,	Product_Supplier	228, 317, 320, 321
833, 834, 825, 826, 827, 828, 829, 830, 831, 832, 833,		products	310
834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844,		Profile_Number	230, 514
845,	846, 847, 848, 849, 850, 851, 852, 853, 854	List	275, 365, 366
problem_scs	498	Profile_Percentage	230, 515
problem_size_and_time (Date_Range)	371	List	357, 369, 677, 678, 679, 680, 681, 682, 683, 684, 686,
problem_size_and_time (Date_Range)	371	687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697,	
problem_size_and_time (Date_Range)	371	698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708,	
problem_time (Date_Range)	371	709,	710, 711, 712
problems	347, 354, 359, 370, 380, 395, 403, 410, 415, 422,	Profile_Quantity	230, 516
process	431, 439, 443, 447, 452, 486, 718, 792	PROMISE	564, 836, 837
process	284, 379, 586, 587, 588, 589, 590, 591, 593, 594,	List	354, 452, 718
605,	606, 607, 608	Promise	229, 352, 355, 392, 411, 413, 414, 415, 416, 423,
process_size	154	438, 562, 716, 719, 721, 723, 724, 726, 728, 729, 731,	
PRODUCE	556, 603	732, 734, 736, 737, 739, 741, 742, 744, 747, 748, 749,	
PRODUCE_FIXED	556, 598, 599	751, 752, 753, 754, 755, 756, 757, 758, 759, 763, 765,	
produce_flow	283	766,	768, 770, 771, 773, 775, 808
PRODUCE_PER	556, 599	promise	392, 438, 721, 723, 724, 726, 728, 729, 731, 732,
produce_separately	870, 871, 872, 873, 874	734, 736, 737, 739, 741, 742, 744, 747, 748, 749, 751,	
produce_time	665	752, 753, 754, 755, 756, 757, 758, 759, 763, 765, 766,	
PRODUCE_YIELD	556, 599	768,	770, 771, 773, 775
PRODUCE_YIELD_CALENDAR	556, 600, 601	promise_as_planned	354, 414, 420, 429, 718
producing_flow	293, 383	promise_by	392
producing_flow (Date_Range)	357	promise_by_policy	420
PRODUCING_FLOW_CALENDAR	557, 558, 623, 624,	PROMISE_DUE	567, 872
625,	626, 627, 628, 627, 678, 679, 680, 681	promise_filter	836
PRODUCING_FLOW_CALENDAR_FILTER_AND_R		PROMISE_NOT_ACCEPTED	559, 564, 754, 755, 834
ANK	557, 558, 628, 629, 630, 631, 632, 633, 634,	PROMISE_NOT_OFFERED	559, 564, 753, 754, 834
635,	636, 681, 682, 683, 684	PROMISE_NOT_PLANNED	559, 563, 728, 729, 822,
producing_flow_plans	357, 364	823	
producing_operation	621, 628, 632, 649, 650, 655, 657,	PROMISE_PLAN	564, 837
658,	664, 666, 667, 669, 671	PROMISE_PLANNED_EARLY	559, 563, 731, 732,
List	310, 331, 332, 332, 339, 340	733,	824
Product	229, 310, 319, 322, 323, 324, 325, 326, 327, 328,	PROMISE_PLANNED_EXCESS	559, 563, 734, 735,
329, 330, 331, 332, 333, 334, 335, 336, 343, 435, 457,		736,	825
459,	555, 580, 581, 583, 584, 585, 713	PROMISE_PLANNED_LATE	559, 563, 730, 731, 823
product	319, 334, 335, 343, 435, 457, 459, 713	PROMISE_PLANNED_SHORT	559, 563, 733, 734, 824,
Product_Allocation	229, 336, 583	825	
product_atp	434	PROMISED	567, 873, 874
Product_Available_To_Promise	229, 434, 435, 436, 437	promised_date_problem	403, 422
product_forecast (Symbol)	451	promised_carry	401, 419
product_forecasts	431	promised_excess	407, 429
List	310, 332	promised_late	401, 419
Product_Group	229, 310, 337, 338, 339, 340, 341, 342,	promised_overpriced	401, 408, 420, 429
343,	344, 457, 459, 713	promised_price_problem	403, 410, 422, 431
product_group	341, 457, 459, 713	promised_quantity_problem	410, 431

## Index

promised_short	407, 428
promises	352, 355, 716, 719
promising_policy	398, 804, 805, 806, 807
promising_policy_attr	417
propagation	788, 789, 816
proper_string	70
Property_List	37
PROPORTIONAL_INTERACTION	561, 563, 790, 818
PROPORTIONAL_RESOLVES	561, 563, 790, 817
0	
quantities	511
QUANTITY	566, 567, 862, 868, 875, 876
List	457
Quantity	37, 248, 249, 272, 297, 298, 299, 317, 324, 330, 334, 335, 341, 343, 354, 357, 360, 361, 363, 369, 370, 371, 377, 384, 385, 407, 409, 428, 429, 430, 434, 436, 446, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 478, 479, 500, 512, 513, 514, 515, 516, 525, 567, 574, 575, 576, 577, 579, 597, 598, 599, 600, 612, 622, 623, 624, 625, 626, 629, 630, 649, 650, 655, 656, 657, 658, 663, 664, 665, 673, 674, 675, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 715, 718, 780, 781, 782, 783, 784, 785, 801, 802, 821, 822, 824, 825, 828, 832, 833, 840, 841, 842, 843, 849, 850, 853, 854, 862, 868, 875, 876, 877, 889, 890, 899, 377, 384, 385, 406, 428, 436, 445, 513, 655, 664, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1130, 1131, 1132, 1133, 1134, 1135, 1136, 1137, 1138, 1139, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1150, 1151, 1152, 1153, 1154, 1155, 1156, 1157, 1158, 1159, 1160, 1161, 1162, 1163, 1164, 1165, 1166, 1167, 1168, 1169, 1170, 1171, 1172, 1173, 1174, 1175, 1176, 1177, 1178, 1179, 1180, 1181, 1182, 1183, 1184, 1185, 1186, 1187, 1188, 1189, 1190, 1191, 1192, 1193, 1194, 1195, 1196, 1197, 1198, 1199, 1200, 1201, 1202, 1203, 1204, 1205, 1206, 1207, 1208, 1209, 1210, 1211, 1212, 1213, 1214, 1215, 1216, 1217, 1218, 1219, 1220, 1221, 1222, 1223, 1224, 1225, 1226, 1227, 1228, 1229, 1230, 1231, 1232, 1233, 1234, 1235, 1236, 1237, 1238, 1239, 1240, 1241, 1242, 1243, 1244, 1245, 1246, 1247, 1248, 1249, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1259, 1260, 1261, 1262, 1263, 1264, 1265, 1266, 1267, 1268, 1269, 1270, 1271, 1272, 1273, 1274, 1275, 1276, 1277, 1278, 1279, 1280, 1281, 1282, 1283, 1284, 1285, 1286, 1287, 1288, 1289, 1290, 1291, 1292, 1293, 1294, 1295, 1296, 1297, 1298, 1299, 1300, 1301, 1302, 1303, 1304, 1305, 1306, 1307, 1308, 1309, 1310, 1311, 1312, 1313, 1314, 1315, 1316, 1317, 1318, 13



SEQUENCE_RUN_ONCE	561, 563, 786, 787, 813, 814	SORT_BY_EXPRESSION	561, 563, 791, 818, 819
SEQUENTIAL_ALTERNATES	561, 563, 788, 789, 815, 816	sort_subst Void_Expression	46
563, .....	789, 790, 816, 817	sort Void_Expression	47
seq_end, Date, Range, Date	132	Sorted_Bucket	229, 363, 364, 677
seq_expression	897	sorted_buckets	677
seq_interval, Quantity, Range, Quantity	106	sorting_criteria	819
seq_max, Quantity, Range, Quantity	105	source	532
seq_min, Quantity, Range, Quantity	105	spec 251, 300, 314, 507, 543, 796, 799, 857, 879, 880, 881,	
seq_option, String, String	152	883, 885, 886, 889, 890, 891, 892, 893, 895	
seq_start, Date, Range, Date	132	specific, forecasts	436
seq_time, Date, Range, Time	133	specific_products	331
selectn, String, String	150	specificfication . 517, 879, 880, 881, 883, 884, 886, 889, 890,	
SHARED_USE	557, 614, 615, 616, 619	892,	893
shifts, Date, Range	138	specifics_allocated	470
shifts, Date, Range, Time	139	specifics_available	474
shifts, Date, Range, Time	140	specifics_committed	466
SHIP_IN_RATIO	562, 807	specifics_consumed	472
short_name	492	specifics_forecasted	465
shortage_quantity	782	specifics_planned	469
shortage_time	782	SPLIT	565, 853
shutdown, String	168	split	336
SIMPLE	567, 897	split (item, Quantity, Logical)	380
SIMPLE_CONSOLIDATION	557, 614, 619	split_multiple	623
SIMPLE_FIXED_QUANTITY	555, 574, 575, 580	splittable	586, 587
SIMPLE_FIXED_TIME	555, 575, 576, 580	sprl_Number	94
sin_Number	94	stable_time	485
SINGLE_REQUEST	555, 575, 580	STANDARD	561, 796, 799
List	233, 238, 313	standard_deviation_demand_calendar	674
228, 233, 235, 237, 238, 239, 240, 241, 242, 244, 245,		standard_deviation_lead_time_calendar	674
246, 249, 253, 254, 259, 263, 269, 273, 274, 276, 287,		start	496
300, 301, 302, 304, 315, 317, 320, 325, 350, 555, 568,		start, Date, Range	133
570, .....	592, 593	start_location	290
List	315, 350	start_of_day, Date	118
List Group	228, 309, 313, 314, 315, 318, 325, 555	start_of_day, Date, Integer	118
309		start_of_month, Date	119
309		start_of_month, Date, Integer	119
309		start_of_week, Date	119
309		start_of_week, Date, Integer	119
309		start_of_year, Date	120
309		start_of_year, Date, Integer	120
347, 350		start_year, Date, Integer	290
372, 374, 380, 386, 388, 393, 396, 399, 411, 415, 437,		STARTED	562, 800, 801
439, .....	559, 716, 719	static_spec	387, 800, 801
static_plan (Plan)	240	std_split	341, 343
static_plans	347	std_time	377
slices	233, 313	sidev_List_Integer	112
size	267, 368, 611, 612, 613	sidev_List_Number	113
size_calendar	613	sidev_List_Percentage	113
Size, Dimension	228, 272, 614	sidev_List_Quantity	113
size_profile (Date, Range)	369	sidev_List_Time	114
size_usage_profile (Date, Range)	369	slcp_	164
sized_capacity (Date, Range)	369	slcp_Integer	165
Skill 228, 243, 270, 273, 274, 275, 276, 289, 561, 569, 797		stocking_policy	256, 671
skill	270, 289	stop	484
Skill_Resource	228, 274, 275, 276, 561, 798	stop_Symbol	163
skills	243, 265, 569	stop_Symbol_Symbol	163
SLIDING	555, 584	stopped	484

## Index

626, 631, 649, 550, 655, 657, 658, 663, 665, 666, 674, 778, 782, 784, 820, 821, 823, 824, 826, 827, 831, 832, 839, . . . . . 841, 842, 847, 852, 863, 869, 878, 891, 892, 588, 589, 590, 591, 593, 863	undense dimensions UNIDENTIFIED_OP_STATE . . . . . 560, 564, 775, 845 unique_List_Void . . . . . 48 Unit . . . . . 230, 253, 258, 285, 319, 340, 511, 512, 513 unit . . . . . 253, 258, 283, 319, 340 Unit_Quantity . . . . . 220, 511, 513
time_calendar . . . . . 589 time_Date_Range . . . . . 134 time_Format . . . . . 893 TIME_MULTIPLE . . . . . 556, 590, 607 time_pad . . . . . 330 time_per . . . . . 588, 590, 591 time_per_calendar . . . . . 589 time_Quantity . . . . . 94 time_String . . . . . 95 time_String_Symbol . . . . . 95	units_of_safety_stock . . . . . 672 units_Quantity . . . . . 96 UNLIMITED . . . . . 557, 611 UNRELEASED . . . . . 559, 564, 745, 829 UNRESTRICTED . . . . . 488 UNRESTRICTED . . . . . 562, 563, 807, 809, 812 UNSPECIFIED . . . . . 566, 868 unspecified . . . . . 518
top . . . . . 303, 332, 338 top_active_strategies . . . . . 348 top_forecastis . . . . . 450 top_item_groups . . . . . 243, 569 top_items . . . . . 243, 569 top_locations . . . . . 243, 569 top_operation . . . . . 800 top_operation_plan . . . . . 379 top_product_groups . . . . . 310 top_products . . . . . 310 top_seller_plans . . . . . 347 top_sellers . . . . . 234 top_site_plans . . . . . 347 top_slices . . . . . 233 total_cost . . . . . 794 total_interaction . . . . . 792 total_interactions . . . . . 793 total_revenue . . . . . 794 total_revenue . . . . . 794 total_shortness . . . . . 793, 794 trace_time . . . . . 153 unavailable_String . . . . . 149 unm left_String . . . . . 73 unm right_String . . . . . 73 unm_String . . . . . 73 unrecoded format . . . . . 896 Type . . . . . 40, 547 Type . . . . . 547 Type_String . . . . . 152 Type_Typed_Value . . . . . 152 Last . . . . . 546 Typed_Value . . . . . 40, 549, 551 typed_value_void . . . . . 150	unwatch_Symbol . . . . . 165 upset_String . . . . . 73 UPSTREAM . . . . . 565, 856 upstream_bucket_pad . . . . . 616 upstream_flow_plans . . . . . 383 upstream_join_flows . . . . . 383 upstream_pad . . . . . 611 usage_policy . . . . . 289, 293, 596, 598, 599, 600 use_alternate . . . . . 376 use_alternate (Operation_Plan) . . . . . 376 USE_ALTERNATE_OPERATION . . . . . 565, 855 USE_ALTERNATE_RESOURCE . . . . . 565, 855 USE_EFFECTIVE_ALTERNATE . . . . . 565, 855 USE_LESS . . . . . 565, 853 USE_MORE . . . . . 565, 855 use_sid_split . . . . . 339, 460, 714 USER . . . . . 567, 897 List . . . . . 533 User 230, 532, 533, 534, 535, 536, 538, 539, 540, 541, 542, 543, 544, 545
user . . . . . 167 user_bias . . . . . 672 user_defined . . . . . 548	user . . . . . 167 user_bias . . . . . 672 user_defined . . . . . 548
V . . . . . 514, 515, 516, 522, 862 value . . . . . 549 value (Typed_Value) . . . . . 153 value_Typed_Value_Type . . . . . 546 values (Typed_Value) . . . . . 166 values_Type . . . . . 266, 610 variability . . . . . 387, 393, 401, 402, 407, 408, 414, 420, 425, 429, 430, 434, 436, 438, 439, 441, 445, 446, 452, 457, 472, 476, 482, 484, 501, 533, 567, 583, 606, 677, 717, 718, 879	V . . . . . 514, 515, 516, 522, 862 value . . . . . 549 value (Typed_Value) . . . . . 153 value_Typed_Value_Type . . . . . 546 values (Typed_Value) . . . . . 166 values_Type . . . . . 266, 610 variability . . . . . 387, 393, 401, 402, 407, 408, 414, 420, 425, 429, 430, 434, 436, 438, 439, 441, 445, 446, 452, 457, 472, 476, 482, 484, 501, 533, 567, 583, 606, 677, 717, 718, 879
UNALLOCATED_FORECAST . . . . . 560, 761 unattach . . . . . 387 UNCONSOLIDATED . . . . . 560, 564, 775, 845 unconsolidated_operation_plans . . . . . 619 UNCOORDINATED . . . . . 560, 564, 776, 846 UNDERLOAD . . . . . 560, 565, 779, 780, 848 underload . . . . . 780 underscore (Symbol) . . . . . 777	IV . . . . . 168 wait . . . . . 168 watch . . . . . 165 watch_Expression_Symbol_Expression . . . . . 865 week . . . . . 810 week_buckets . . . . . 124 week_of_year_Date . . . . . 777

week_periods	311
weekday_format	894
WEEKDAYS	566, 863
WEEKENDS	566, 863
WEEKLY	555, 577, 578, 580
WEEKS	562, 810
weeks_Date_Range	141
weeks_Date_Range_Integer	142
WEIGHTED_LATENESS	561, 566, 793, 859
WEIGHTED_SHORTNESS	561, 566, 793, 860
where	165
whereis	165
while_Logical_Void	160
width	517, 895
wildcard_String_String	76
with_connection_Connection_void	135
with_echo_file_String_void	156
within(Location)	247
within_Date_Date_Range	134
within_Date_Range_Date_Range	135
within_daylight_savings_time_Date	125
within_leap_year_Date	125
within_Quantity_Quantity_Range	98
within_Quantity_Range_Quantity_Range	99
Worksheet	230, 534, 540
worksheet	534
X	
x	517
x_position	248
x_size	248
x_size_min	248
xor_Logical_Logical	144
Y	
y	517
y_position	248
y_size	249
y_size_min	248
year_Date	124
YEARLY	566, 867
yield	599, 600
yield_near	599, 600
Z	
ZERO	557, 610, 611
zero_available_io_promise	476